MASTER IN COMPUTER SCIENCE

Electronic mail gateways

Introduction to the state-of-the-art an description of a particular implementation : CERN

Derruine, Eric

Award date:
1988

Awarding institution:
University of Namur

Facultés Notre Dame de la Paix, Namur
Institut d'informatique

Année académique 1987-1988

ELECTRONIC MAIL GATEWAYS

Introduction to the state-of-the-art and
description of a particular implementation : CERN

Eric Derruine

Promoteur : Philippe van Bastelaer

Mémoire présenté en vue de l'obtention
du titre de licencié et maître en informatique

Acknowledgements

The realization of this work followed a six month traineeship at the European Organization for Nuclear Research (CERN) at Geneva. I would like to thank Professor Philippe van Bastelaer who trusted me and gave me the opportunity to live this formidable experience. With great attention did he also supervise the redaction of this dissertation.

I would like to thank the people at CERN too, who welcomed me and integrated me in their own work. With them I learned lots of things that are not always easy to teach at university.

Denise Heagerty helped me understand what electronic mail is all about. She is also the main source of information for this work. Olivier Martin introduced me to the IBM world, as well as to the Telex network. Giorgio Heiman let me know DECnet and the VAX/VMS machines.

Other staff members at CERN helped me one way or the other. In particular, François Fluckiger, Brian Carpenter, Tor Botner, Marie-Thérèse Monnet and all the Telex office people.

I also would like to express special thanks to Dietrich Wiegandt who very kindly read and commented most of this study.

Table of contents

List of figures

List of tables

Introduction

Computer networks began to appear about twenty years ago (the ARPANET project started in 1969). Since then, communication needs between machines ranging from personal computers to supercomputers have grown more each year.

The main problem with networks is that they were developed independently from each other, using different architectures and protocols. At the beginning, this was not a major problem since computer networks were initially set up to respond to specific needs related to limited communities.

However, this is not true anymore. More and more services are offered by computer networks, including file transfer, remote job execution, remote login and electronic mail (E-mail) and most of the time, these services are incompatible from one network to the other. Moreover, the utility of a network as well as the benefits its subscribers can draw from it are directly linked to the number of accessible machines and users.

The need to link the existing networks in order to increase the global connectivity emerged quite soon. A lot of work has already been done in the field of computer interconnection, but the problems are numerous and not easy to solve. The general problem has been simplified by using layered networking models (like ISO's Open Systems Interconnection model), and by approaching the interconnection issues one layer at a time.

However, the migration by existing networks to these international standards is a slow process and great care must be taken to ensure that the transition happens as smoothly as possible. When everyone will conform to these standards, connectivity and incompatibility problems will have disappeared.

In the meantime and in parallel with a migration towards international standards, it has been necessary to set up gateways to ensure the greatest connectivity between existing networks and services. However, the task of interconnecting the different layers of network architectures has proved to be more complicated when proceeding towards higher layers. Interconnection at the network layer (layer 3 of the OSI model) is in good way. Interconnection of higher levels, however, has started only quite recently.

One of the first network services (application layer of OSI) of which the interconnection is being tackled is E-mail. Generally speaking, E-mail allows persons to exchange electronically all the information that they would otherwise exchange by conventional mail (letters, audio and video cassettes,...) or by telephone. This includes the sending of text, graphics, voice and moving images but excludes services like file transfer or remote job entry. Most networks are using different E-mail systems for their internal communications.

In 1984, the CCITT accepted the X.400 international standard specifying a Message Handling System model and all the related protocols. Most networks are currently planning to migrate to X.400 for their E-mail systems. While waiting for the perfect world where everyone will talk X.400, mail gateways are to be set up between the existing incompatible E-mail systems to allow users on one network to exchange messages with users on other networks.

This work is an introduction to the problem of interconnecting E-mail systems via E-mail gateways and to the state-of-the-art in that field. It was realized following a six month traineeship at the European Laboratory for Particle Physics (CERN), Geneva, where an E-mail gateway system between several major networks is operated. This explains that a lot of references will be made to CERN's implementation and CERN's gateway managers, which are the main sources of information for this study.

The structure of this work is the following. Chapter 1 describes the X.400 Message Handling System. This will help to understand the components of E-mail systems and will provide a reference model. Chapter 2 is an overview of the major networks existing throughout the world, their E-mail systems and the protocols they use. The functions required from an E-mail gateway are described in chapter 3. Chapter 4 outlines the problems that arise following the introduction of E-mail gateways. Gateways have to be looked after by some benevolent persons, the gateway managers. Chapter 5 will describe the tasks of these. Since it provided the basis for this work, the implementation of the gateway system set up at CERN will be detailed, in chapter 6. Chapter 7 will describe a tool implemented by the author during his traineeship at CERN and aimed at helping the gateway manager in one of his tasks (namely the analysis of log files).

A list of the abbreviations used throughout this study can be found in appendix 1. Appendix 2 will contain the listing of the set of programs and data files implementing the tool described in chapter 7.

Chapter 1
The X.400 Message Handling System

Trying to interconnect different E-mail systems requires at least a good understanding of their components and of the details of the functioning of the latter, when taken separately. Since it would not be possible to describe in detail all the E-mail systems mentioned in this study, and also because the descriptions of most of these would be redundant because of the inherent similarities, the decision was made to describe only one of them.

Choosing a representative E-mail system was not too difficult. As a matter of fact, there exists an E-mail system commonly regarded as the future standard towards which all current implementations are heading : the CCITT X.400 series of recommendations. The E-mail system described by those recommendations is also known as the X.400 Message Handling System (MHS).

The X.400 MHS is the obvious choice when the goal is to explain the working of E-mail systems in general, for several reasons. Firstly, it has benefited from the experience gained with previous and current practical E-mail systems, and in this regard, it features the main (and most valuable) characteristics that can be found in real systems.

Secondly, it is well known and well understood. There have been lots of papers written about it. Most of the specialists in the field consider it as the model best representing the state-of-the-art in E-mail today.

And last but not least, there is the fact that it is considered as THE standard, by international standard organizations which fostered its development (IFIP, ISO, CCITT) and also, what is far from being worthless, by most manufacturers who are developing E-mail system implementations conforming to the X.400 MHS.

The X.400 recommandations series was released in 1984, but the 1988 version is already announced, with non negligible differences with respect to the previous one.

1.1 The Message Handling System model

The model described by the X.400 recommendations is a hierarchical model, whose lowest level is the Message Transfer System (MTS). The MTS is composed of Message Transfer Agents (MTA) whose main function is to relay messages through the MTS.

The upper level is composed of User Agents (UA) which provide a kind of interface between the MTS and the users they support (there is one UA per user). On the one hand, a UA helps its user to build and submit messages to the MTS, and on the other hand, it receives the messages delivered by the MTS and helps its user to interpret these messages. A UA can offer other services as well, as for example the storage and retrieval of previous messages, sophisticated editing and presentation of the messages, etc.

Messages transferred by the MTS between UA's are composed of an envelope, containing mainly the addresses of the origin and the destination of the message, and a content, which is normally left untouched by the MTS.

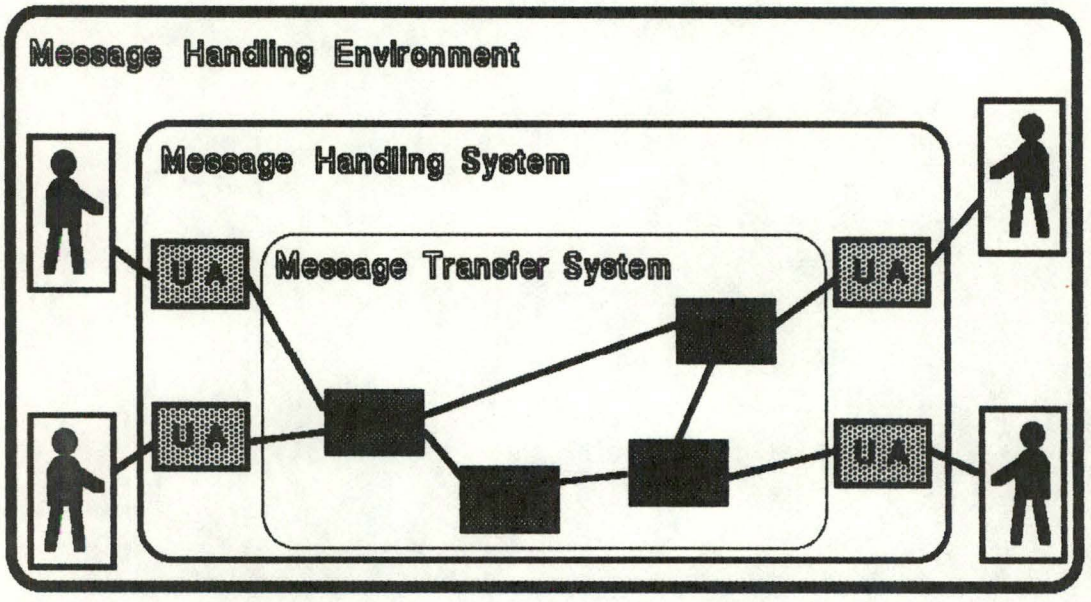


Fig 1.1 : The Message Handling System (MHS) model

Figure 1.1 shows the representation of the MHS model, with, from the inside to the outside, the MTS containing only the MTA's, then the MHS containing the MTS plus

the UA's, and finally the whole Message Handling Environment, containing the MHS plus the users.

1.2 The Message Transfer System

A UA is associated to one and only one MTA. The main interactions between UA's and their respective MTA's are the submission and the delivery of messages. An MTA first accepts a message submitted by an originator UA, forwards it to the next MTA towards the destination and so on until the message arrives at the MTA of the destination UA, which delivers it.

It must be noted here that the policy of the designers of the MHS was to rely on a store-and-forward service, instead of on a connection-oriented point-to-point service, making no assumption of real-time connection between the origin and the destination of a message. The main drawback of a point-to-point connection service is indeed its unability to deliver a message unless all the entities corresponding to MTA's between the origin and the destination are available at the precise moment of the transfer of the message.

The MTS is used by specifying which service element it is requested to provide. Among other things, these service elements allow a UA to establish the communication with its MTA and to give to the latter all kinds of information concerning the message to be delivered. They also enable a UA to ask its MTA for delivery or non-delivery notifications, status information, conversion of messages and even to use a probe to know if a given correspondent is reachable or not.

1.3 The InterPersonal Messaging System

The MTS was designed to be able to support any kind of connectionless communications. In order to use it for interpersonal messages, i.e. between individuals as opposed to computer processes, a set of rules was defined by the CCITT, specifying what is called the InterPersonal Messaging System (IPMS).

An example of UA's not being part of the IPMS is a group of UA's using the MTS only to periodically communicate the results of local processes of the branches of a same company to a process running in a host at the company's main computing center.

The IPMS comprises the MTS and a specific class of cooperating UA's which use a protocol of their own (called P2) to communicate. Two special protocols were also set up to enable the IPMS UA's to communicate with Telex and Teletex users. Fig 1.2 shows a representation of the IPMS model.

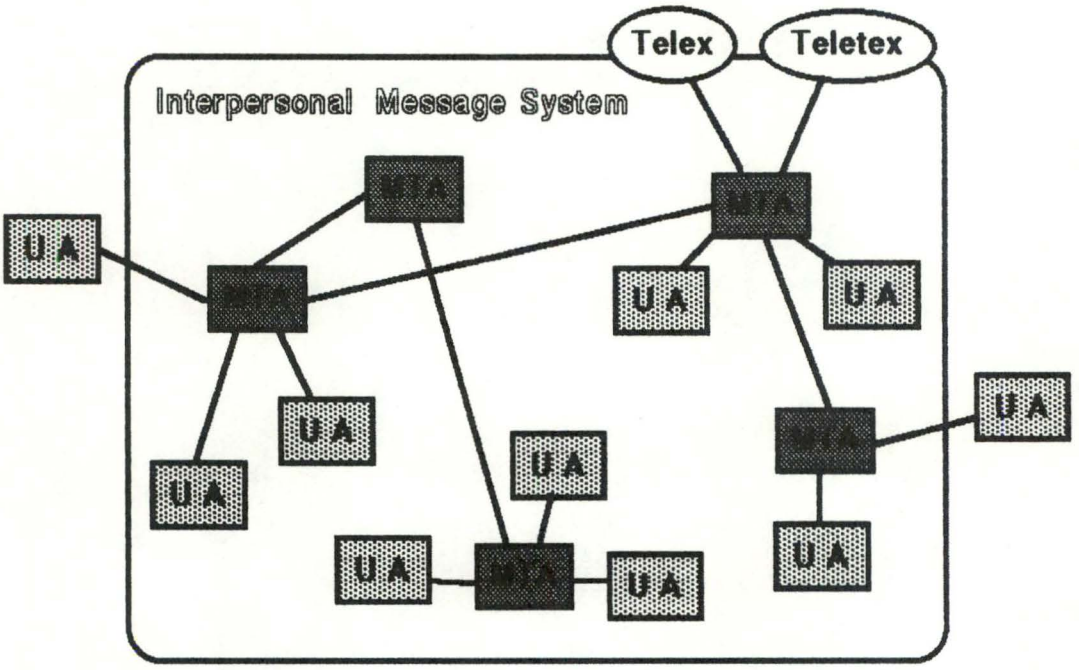


Fig 1.2 : The InterPersonal Messaging System (IPMS) model

The structure of an interpersonal message is further refined with regard to the envelope-content structure, as shown in figure 1.3. The content part is composed of a heading part and a body part. The heading part will contain information as the names of the originator and the addressee(s), the names of secondary addressee(s), the date, the subject of the message and so on. The body part will possibly be composed of several parts, which could include, in the future, text, facsimile, graphics, videotex and even voice information (but these options are not yet fully specified and thus are left for further study).

Moreover, each interpersonal message has its own identifier, independent of the MTS message identifier, by which it can be referenced unambiguously, in later correspondence for example.

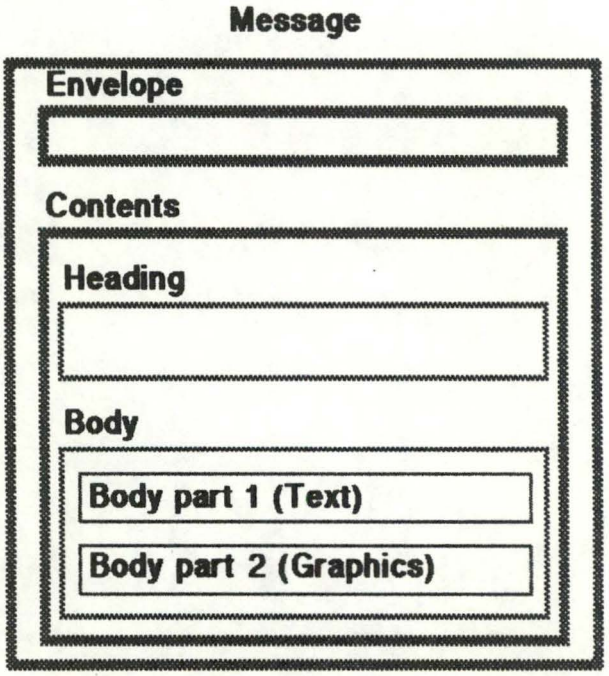


Fig 1.3 : Example of interpersonal message

1.4 Layered representation of the Message Handling System

The MHS model can be given a layered representation (see fig 1.4), which is well a known concept thanks to ISO's Open Systems Interconnection (OSI) reference model. There is first the UA layer, composed of the functional entities corresponding to the UA's (UA entities).

Then, there is the Message Transfer Layer (MTL), composed of MTA entities which provide the functionalities required to support the services offered by the MTL, in cooperation with other MTA entities. The MTL also comprises another kind of functional entities, the Submission and Delivery Entities (SDE), which make the MTL services available to UA's which cannot interact directly with their own MTA because they are not located on the same physical system.

These two layers are in fact sublayers of the application layer as defined in the OSI model.

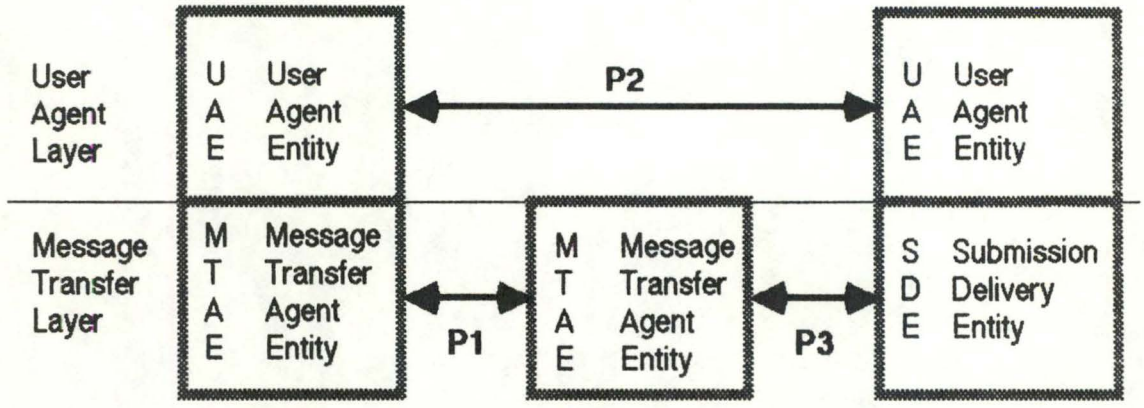


Fig 1.4 : Layered model of the MHS

It should be noted here that the different functional entities do not have to be located in the same system. The different configurations that we can find in a single system are the following. Firstly, it's possible to find a UA and its MTA in the same system, in which case they interact directly.

Then, a UA can be found on system physically different from the one of its corresponding MTA, in which case it communicates with the latter via an SDE. Such a stand-alone UA (with it's corresponding SDE) could be implemented in a personal computer, intelligent terminal or equivalent.

Finally, it's also possible to find an MTA standing alone, with no UA's on the same system, in which case this system can only act as a relaying site for the MTS.

Different protocols enable the different functional entities to communicate with their peers. Protocol P1 defines the relaying of messages between MTA's and other interactions necessary to provide MTL services. Thus, P1 is a protocol dealing only with the transfer of messages. This type of protocol will be called a "transfer protocol" in the rest of this study.

The choice of the protocol used between cooperating UA's is not compulsory, since different types of protocols correspond to different classes of UA's. However, in the particular case of the IPMS, the X.400 recommendations specify such a protocol, called P2. These protocols deal only with the content of messages and must be carefully distinguished from the transfer protocols. This type of protocol will be referred to as an "end-to-end protocol" or "content protocol" in the following.

Content protocols are very important since they are the ones that determine what kind of functionality will be offered to the user. For example, some content protocols will allow a user to request a confirmation that their message was actually delivered to the intended addressee (e.g. P2), while others won't (e.g. RFC 822, which will be described in section 2.3.1).

The submission and delivery protocol, P3, allows an SDE to communicate with a remote MTA, enabling the UA it supports to have access to the MTS.

1.5 The Management Domains

The MTA's are grouped in management domains which own and control them. These domains can be managed by a public administration, in which case we speak of ADministration Management Domains (ADMD), or by private organizations, in which case they are called PRivate Management Domains (PRMD).

A UA can be part of the management domain of its MTA if it is provided there (i.e. hired to the subscriber), or it can be outside the management domain, if it belongs to the subscriber and is situated in an intelligent terminal or in a personal computer.

ADMD's represent that portion of the MHS under control of public service providers such as GTE Telenet and Tymshare in the USA, and the PTT's in other countries. PRMD's represent those parts of the MHS under control of organizations such as corporations that provide in-house message service to limited, privately selected groups.

A management domain (PRMD or ADMD) may not span several countries. A PRMD may not act as a relay between ADMD's.

1.6 Naming and addressing

One of the basic objectives of the designers of the MHS was the ability to address people by name and other personal attributes, rather than only by terminal address.

Consequently, the X.400 recommendations specify two different ways to reference a user (or rather his associated UA).

The first means is the use of a set of user attributes by which recipients and originators can be identified. This set of attributes is called the Originator-Recipient name (O/R name).

These attributes include personal attributes (e.g., surname, given names, and initials), geographical attributes (e.g., country), and organizational attributes (e.g. company name and organizational unit, such as division or department).

The second means is to use an O/R address, which is a descriptive name for a UA that has certain characteristics that help the MTS to locate the UA's point of attachment. In fact, an O/R address is also a special kind of O/R name.

To specify an O/R address, architectural attributes are used. These include X.121 addresses, unique UA identifiers (numeric value), and specification of the ADMD or PRMD.

Normally, in order to reference his addressee, a user would specify the O/R name of the latter. But since an O/R address is also an O/R name, it is possible to specify the addressee directly by his O/R address, if the latter is known. This has the advantage of being more cost-effective, since the directory look-up necessary to map an O/R name to an O/R address is bypassed. But an O/R address has the drawback of being less easily remembered and of being more likely to change than an O/R name.

Moreover, the use of O/R names requires the establishment of directories that can identify either a single individual based on the specified set of attribute values or the next MTA to which the message should be relayed. But only when standard protocols for directory coordination have been defined can pure names be used. Otherwise, some form of addressing information implying location is required to route messages. As the specification of directory protocols could not be completed soon enough by the CCITT, it was decided to add for an interim period one additional attribute identifying the Management Domain associated with a given O/R. This attribute facilitates the task of the MTS when the latter has to route a message, by making it unnecessary to look in some directory which is the Management Domain associated to the recipient's O/R name, as will be necessary when pure logical O/R names will be used.

In the future, the X.500 standard is expected to solve most of the problems related to general directory services and protocols.

It is also worth noting that a special domain defined attribute has been specified to permit the identifiers assigned by existing systems to be used. The intent is to ease the transition to standard naming conventions. One of the best known use of this possiblility is illustrated by the way the EAN E-mail system has chosen to code addresses to ease gatewaying to the world of E-mail systems complying to the RFC 822 standard, as will be seen in section 2.4.4.

Chapter 2
Major computer networks and their E-mail systems

Now that we have described a model of electronic mail system, we are in a bettter position to have a look at what really exists and to make a description in the terms of the model. Some of the major networks currently used will be briefly described (section 2.1), as well as their corresponding E-mail systems. These will be more specifically decomposed with respect to their transfer (section 2.2) and content (section 2.3) protocols, which will be described. There will also be an overview of the different addressing schemes (including mail address formats) used in the different networks (section 2.4).

The reason for having a closer look at each of the main components of these E-mail systems is that it is precisely those E-mail systems that we are willing to interconnect via gateways. It is thus important to know well their different components, with respect to the interconnection process.

2.1 Overview of some major networks

The term computer network is used to describe an entity with three major components [Landweber86] :

- a collection of host computers or hosts, which provide computing services to users
- a communication subset, which consists of special purpose communication processors, called nodes, switches or Interface Message Processors (IMP's), connected by some communication medium (telephone lines, dedicated leased land lines, radio channels, coaxial cables, or satellite channels)
- a set of communication protocols

The networks briefly described here were chosen for several reasons, some of which are specific and will be discussed separately in the section relative to each network, some of which are common to all and are discussed hereafter.

First, they all have currently a certain importance in terms of number of hosts and users supported worldwide. They also are particularly important because all except the ARPA Internet will be directly involved in the gateway system described in chapter 6.

For more details about these and other networks, see [Quaterman86].

2.1.1 ARPA Internet

The ARPA Internet is the main network in the USA and is one of the biggest in the world. It is also very important because the protocols developed and used by it for different purposes constitute standards even for other networks (SMTP and RFC 822 as mail protocol, FTP as file transfer protocol, TCP/IP as transport/internetwork protocol, Telnet for remote login,...).

The ARPA Internet is an internetwork of several networks all running the TCP/IP protocol suite, connected through gateways (at the network level) and sharing common name and address spaces. Its name comes from the Defense Advanced Research Projects Agency (DARPA), which has long been a major sponsor of networking research in the USA.

ARPANET and MILNET, which are the two principal backbone networks of the Internet, are funded mostly by government grants. Others are funded by local organizations.

The Internet hosts are connected by 56 Kbps dedicated lines and a few satellite links. Reliability is considered very high. The service offered include file transfer (FTP), remote login (Telnet), mail and others. The mail system uses RFC 822 as content protocol and SMTP as mail transfer protocol.

The ARPA Internet is a research network, which means that its objectives are to share resources among its members, to ease communication between these and to provide a testbed for networking research. Practical coordination of the entire Internet is provided by the Network Information Center (NIC) at Stanford Research Institute (SRI) International and the Network Operations Center (NOC) at Bolt, Beranek and Newman (BBN).

The ARPA Internet is organized in a domain name system. Names servers distributed throughout the Internet are used to determine how to reach any host.

A very interesting feature of the ARPA Internet is the way standards are designed. All begins with a substantial proposal for a special topic and a request for comments (RFC) about this proposal, which are broadcast through all the Internet. Comments come back to the source of this kind of network-wide conference. The initial proposal is consequently modified and resent through the network. This process eventually stops when there is a general consensus about the proposal, which is then adopted as a new standard. Most standards in use on the ARPA Internet, and even elsewhere, were designed that way. The text of these is available on files and easily accessed, e.g. via file servers on the Internet and have the generic name of RFC xxx.

For example, here is the way the RFC 822 standard was produced. First, the specification of RFC 733 (defining a standard for ARPANET text messages) took place over the course of one year, using the ARPANET mail environment itself. More than twenty persons from across the USA were involved in an on-going forum for discussing the capabilities to be included. RFC 822 was then specified from RFC 733 to take into account the difference between the ARPA Internet and the isolated network ARPANET, still using network mail-based group discussions. Both specification efforts greatly benefited from the comments and ideas of the participants.

2.1.2 BITNET and EARN

BITNET (Because It's Time Network) is a cooperative network linking mainly universities and several research centers in the USA and Mexico. The access to it is virtually unrestricted for academic institutions and there is no membership fee. The only costs incurred are those of the leased line that has to be acquired to facilitate the connection to another BITNET node. This means, among others, that there are only fixed costs, and no volume dependent charges.

BITNET was originally built around IBM machines and software. Now, more and more vendors and members of the BITNET community offer emulation software enabling other machines than IBM's to join the network (e.g. UREP on VAX/Unix and JNET on VAX/VMS). The basic communication protocol used on BITNET is the VM based RSCS (Remote Spooling Communications Subsystem) protocol, usually running over 9600 bps leased telephone lines.

User, technical and administrative support is provided by the BITNET Network Support Center, which is operated jointly by two major BITNET hosts.

The services offered include E-mail, file transfer and interactive messages, which allow two or more users on any host of the network to communicate in nearly real-time. There is no real remote login facility.

Several mail systems exist on BITNET. For example, there is NOTE, which is an IBM product. The NOTE command is fine for sending mail to BITNET sites, even if it doesn't use a recognized standard and suffers from a lack of functionality. However, it can only be used to send mail (another program is necessary to read incoming mail) and it cannot reply automatically or forward messages.

For these reasons, the MIT/Rice Mail Exec, in conjunction with the Columbia Mailer, is increasingly used. It provides BITNET users with a more satisfying mail system using RFC 822 as content protocol and BSMTP (a batch version of the ARPA SMTP protocol) as transport protocol.

BITNET has two counterpart networks, EARN and NETNORTH, respectively covering Europe and Canada. The distinction between them is purely geographical, the services offered being mostly the same and compatible. The design and underlying philosophy is basically identical as well. These three networks total more than 2000 hosts when taken together. A map of all these hosts is sent regularly to all hosts on the three networks and is used to find the unique route between any two hosts.

EARN, the European Academic Research Network, was born in 1984 and was supported by IBM which funded it till the end of 1987. There is one backbone node per country, responsible for administrative and technical support.

2.1.3 UUCP, USENET and EUNET

UUCP is a cooperative network linking machines ranging from personal computers to supercomputers and possibly totalling more than 10000 hosts throughout the world. UUCP (Unix to Unix CoPy) is in fact a transport service constituted of programs distributed as part of the Unix operating system.

Hosts on UUCP are usually connected by 1200 or 2400 bps dial-up lines. Each host pays for its own links but there is no membership fee. The only requirement to connect to UUCP is to find a UUCP host that will agree to be a neighbour.

It is necessary to emphasize here the difference between UUCP and networks like BITNET or the Arpa Internet, which use dedicated (and thus static) lines. UUCP host-

to-host connections, on the contrary, are dynamic. This has the major advantage of being very flexible. All one needs to get in touch with another host is to agree with it and simply to call it by phone.

The services offered are mainly file transfer, remote command execution and mail. There are several mail user interface programs. One of them is the Unix mail package, distributed with Unix 4.2 BSD. In conjunction with the sendmail facility (see section 6.5.1), it provides access to UUCP as well as to other mail networks.

Mail is transferred by submitting a mail command over a direct connection by the UUCP remote command execution mechanism. The arguments of that mail command indicate whether the mail is to be delivered locally on that system or resubmitted to another system. RFC 822 is normally used as the contents protocol.

One of the features of the original UUCP mail is that it uses source routing, meaning that in order to send mail to a user on another host, one has to specify the list of intermediate hosts needed to reach the destination host.

There is no central administration, but a group of volunteers known as the UUCP Project maintains a map of all UUCP hosts. That map is sent regularly on USENET, which is a news network closely related to UUCP, and is used as input to a program (pathalias) to compute reasonable routes between any two registered hosts. Thus, complete path specification has become unnecessary when mail can be sent via such intelligent hosts. Moreover, the widespread use of more sophisticated mail relay programs (such as sendmail and MMDF) has increased reliability.

It is planned to implement naming in the style of ARPA Internet domains and even possibly to include the UUCP name space into the ARPA Internet domain name space.

EUNET (European Unix NETwork), the counterpart of UUCP in Europe, started in 1982. The software and protocols are basically the same but there are still marked differences between UUCP and EUNET. X.25 lines are usually used to link hosts, instead of dial-up telephone lines. Mail and news are much more closely tied in EUNET. The administration is also much more organized. The old-style bang path addresses have almost completely been replaced by the RFC 822 addressing scheme. There is a backbone consisting of one host per country responsible for the national organization of the network (prlb2 in Belgium), and there is also a central host (mcvax in Amsterdam) to which all backbone nodes have direct connections, and which is responsible for intercontinental traffic.

Funding is also different on EUNET : communication costs are more equally shared among the member hosts.

2.1.4 X.400

X.400 is not really a network in the sense taken so far. It is rather a set of computers implementing the Message Handling System as defined by the X.400 recommendations. Today, more and more manufacturers are working on X.400 implementations. Some of these are already running, many others are still in development.

EAN (Electronic Access Network) is the first implementation of the X.400 recommendations and is also the most widely used X.400 product today. It has established a kind of implementation standard, since lots of things were left for further study in the X.400 recommendations, so that implementation decisions had to be taken. Current implementations exist for VAX/VMS and Unix machines.

The first version of EAN was developed at the University of British Columbia (UBC) for Canada's research network, CDNnet, from 1981 to 1983, thus even before the final standard was released, which may explain that it was not totally X.400 compliant.

Mail is the main service provided but other services are also offered. For example, EAN allows in some cases the use of a directory service to locate people on the network (this is implemented on CDNnet, not really on the European EAN networks). USENET news is also available on at least part of the network and remote login is theoretically possible on hosts with X.25 as network layer service.

The objective of the European EAN networks is to establish communication links for the European research community, in cooperation with RARE. RARE (Réseaux Associés pour la Recherche Européenne), is a metanetwork aimed at the unification and standardization of the European national networks, using ISO protocols, and is partly sponsored by the European Community Commission. The are currently about a hundred sites member of RARE which use EAN.

Other X.400 systems currently running are DFN-EAN, KOMEX and GIPSI. DFN-EAN has been developed for the German research network by the Gesellschaft fur Mathematik und Datenverarbeitung (GMD), on Unix systems [Magedanz87]. KOMEX was also developed by GMD. It started as an non-X.400 MHS system but later evolved to X.400. Since 1987, it is in operation as a real X.400 system. GIPSI was developed

by INRIA (French Institut National de Recherche en Informatique et en Automatique) in 1985.

2.1.5 DECNET

In contrast to other networks, DECNET (Digital Equipment Corporation NETwork) is not a unique unified network. Rather, a DECNET is a network of machines based on a network architecture called DECnet, which is proprietary of DEC. DECnet is used to link mostly VAX/VMS machines that want to be on the same network because their users share the same interests. This in fact implies that there is no single DECNET, but rather several DECNET's which do not necessarily communicate with each other.

Two of the most known such networks are the US Space Physics Analysis Network (SPAN) and High Energy Physics (HEP) DECNET's, also called respectively SPAN-NET and HEP-NET.

The services offered by DECnet include file transfer, remote login, remote task execution and E-mail transfer. The E-mail system used is called VMS Mail and is shipped with the VMS operating system. It is the standard mail system for communication with local VMS users or across DECnet. VMS Mail uses a rather primitive end-to-end protocol and DECnet as transfer protocol.

2.1.6 Others

There are of course many other networks, most of which are much smaller than the ones briefly described here. Even though it is not possible (and even not really interesting with respect to our purpose) to talk about each of them, it seems worthwhile mentioning two of them.

The first one is CSNET (Computer Science NETwork). It is a metanetwork built on top of several physical networks, including ARPANET. The hosts of CSNET are mostly located in the USA. Originally, CSNET was designed to provide E-mail services only, mainly to link those who had ARPANET connections to those who did not. E-mail is still the only service provided throughout all the network, even though some parts of CSNET support other kind of services.

JANET (Joint Academic NETwork) was originally set up to link large university computer centers and research establishments in the United Kingdom. The high level

protocols used by JANET were developed by the UK Joint Network Team (JNT) and are globally called the Coloured Book : Blue Book for file transfer, Grey Book for E-mail services, Yellow Book for network independent transport service... Grey Book is based on RFC 822. Moreover, a Name Registration Scheme was adopted to provide a global hierarchical name space like the one used on the ARPA Internet. The domain ordering chosen by JANET is unfortunately "backwards" with respect to the one used by all other networks.

2.2 Overview of the major transfer protocols

Mail transfer protocols are responsible for the physical transfer of messages between hosts on a same network. Two main techniques can be used.

On the one hand, a store-and-forward protocol transfers a message by relaying it from host to host on the route between its origin host and its destination host. If a connection to the following host on the route is not available at a given moment, the actual transfer is postponed and automatically retried later.

On the other hand, a connection-oriented protocol needs to establish a direct connection between origin and destination to be able to transfer a message. The advantage is of course that when this procedure succeeds, one can be sure that the message arrived safe and well after only a few seconds. The big drawback is that if one of the intermediate hosts on the route is not available (i.e. temporary lack of resources, host down,...), then the transfer fails altogether and has to be retried manually, i.e. by the user himself, at another time.

2.2.1 SMTP

SMTP stands for Simple Mail Transfer Protocol. It was originally developed for the ARPANET and is fully specified in RFC 821 [Postel82] and is currently the de facto standard for the ARPA Internet.

The objective of SMTP is to transfer mail reliably and efficiently, either directly between two hosts or by using relay SMTP-servers when there is no direct link between origin and destination hosts.

SMTP is independent of a particular transmission subsystem. All that is required to support it is a reliable ordered data stream channel, like TCP/IP or the ISO TP4 transport protocols, allowing two processes on two different hosts to communicate.

Figure 2.1 describes the SMTP model. Following a user mail request, the sender-SMTP establishes a two-way transmission channel to the receiver-SMTP, using the services provided by its supporting transmission subsystem. The receiver is either the final receiver of the mail or only a relay. The sender-SMTP then generates SMTP commands to the receiver-SMTP which sends replies back. The file system is used to support the mailboxes of the users and to store the information that has to be remembered in order for SMTP to retry later, for example when a connection has failed.

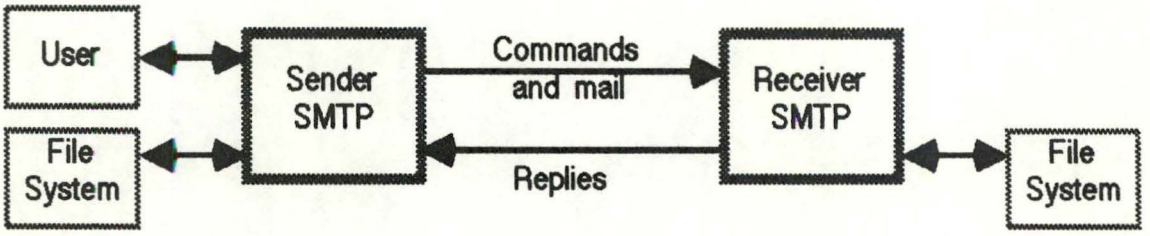


Fig 2.1 : The SMTP model

The usual scenario is the following. After having established the two-way connection to the receiver-SMTP and introduced itself with the HELO command, the sender-SMTP sends the MAIL command, with an argument indicating the origin of the mail it is going to transfer. In fact, this argument specifies a reverse path which could be used by the SMTP system to report errors or by the addressee to reply to the message.

If the receiver-SMTP is able to process the request (i.e. sender-SMTP known, enough resources,etc), it sends back an OK reply. The sender-SMTP then sends one or several RCPT commands, to indicate the recipients of the mail. The argument of each RCPT specifies a forward path which constitutes a source routing from here (i.e. the current relay host, not necessarily the origin host) to the destination host.

For each RCPT command received, the receiver-SMTP sends back either an OK reply, or an error code explaining what is wrong with this particular recipient (i.e. unknown local recipient, unknown host,etc). The sender-SMTP then sends the DATA command, warning the receiver-SMTP that the data is going to be transmitted. If the receiver-SMTP is ready to accept it (i.e. enough resources, at least one valid recipient,etc), it

sends an OK reply. The data constituting the mail message is then transferred, after which the receiver-SMTP sends an OK reply, if all went well.

The data transmitted after the DATA command in fact comprise the headers of the mail message (date, to, from,...) as well as the text itself of the message, which must be composed of 7-bits ASCII characters (this means that the only kind of data that can be transmitted is plain text). The end of the transmitted data is signalled by a special sequence composed of one dot character between two carriage return/linefeed sequences ("<CRLF>.<CRLF>").

A transparency procedure assures that this special sequence will not interfere with the content of the message being transferred, by adding systematically an extra dot at the beginning of a line which starts with a dot. The line is then safely transmitted and the extra dot is removed, keeping the content of the line intact.

Figure 2.2 shows an example of this scenario. User userA on hostX tries to send a message to userB and userC located on hostY. Both hosts are on the ARPANET. We assume that there is a direct link between hostX and hostY, and that userC is unknown on hostY. The character strings "S:" and "R:" respectively indicate the command sent by the sender-SMTP (on hostX) and the reply of the receiver-SMTP (on hostY).

Commands / Replies dialog	Comments
R: 220 HostY.ARPA Simple Mail Transfer Service Ready	Connection establishment
S: HELO HostX.ARPA	The sender introduces itself
R: 250 HostY.ARPA	The receiver replies its own name
S: MAIL FROM:<userA@hostX.ARPA>	-I have mail from userA
R: 250 OK	-All right !
S: RCPT TO:<userB@hostY.ARPA>	-One of the recipients is userB
R: 250 OK	-All right !
S: RCPT TO:<userC@hostY.ARPA>	-Another one is userC
R: 550 No such user here	-Sorry, userC unknown on hostY
S: DATA	-Here comes the message
R: 354 Start mail input; end with <CRLF>.<CRLF>	-Ok, I'm ready. Go ahead.
S: headers	This is the text of the message
S: ... here is my message ...	Yet other lines...
S: <CRLF>.<CRLF>	This is the end of the message
R: 250 OK	
S: QUIT	-Worked enough for today
R: HostY.ARPA Service closing transmission channel	-Ok, see you !

Fig 2.2 : Example of SMTP procedure

The example shows how SMTP respects a design rule common to most protocols on the ARPA Internet. These protocols have indeed been designed for use both by automated processes and by humans. The codes beginning each command or reply are easily processed by machines while the comments following the codes are quite clear and meaningful for a human reader. These protocols are thus usable either as is, directly by a human user, or rather by a higher level user-interface, as is generally the case.

It should also be noted that SMTP uses the transmission subsystem supporting it very efficiently, since it allows several recipients for the same message. Several copies of the message are thus only generated when the routes to reach the different recipients part from each other.

2.2.2 BSMTP

Using SMTP is very handy on networks like the ARPA Internet, which makes use of direct 56 Kbps TCP/IP lines between hosts, but sometimes, it is not possible to use such an interactive protocol. For example, EARN/BITNET and UUCP/EUNET only provide RSCS and UUCP as transfer system, and these are store-and-forward systems where a file is the smallest unit of transmission. So, a full duplex transaction as described by the SMTP procedure is clearly not possible [Crosswell82].

This is a shame, since a protocol like SMTP supported by ARPA is likely to be well defined and widely accepted and understood. BSMTP (Batch Simple Mail Protocol) is one answer to this problem.

BSMTP is a batch protocol simulating SMTP by using separate files for whole sections of a transaction rather than for each command. A sender-SMTP builds a file of commands, just like in SMTP, assuming the reply to each command would be OK. Then, using the store-and-forward system at disposal, the command file is sent to the receiver-SMTP, where each command is interpreted and acted upon, and where a log file containing the usual SMTP responses to the commands is created. This log file is then sent back to the sender-SMTP, where it is analyzed. The results of the analysis are finally reported to the user who originated the mail request, in order for him to take appropriate action (e.g. the log file indicates that one of the recipient could not be reached).

A problem arises immediately. Indeed, how is it possible to be sure which is the command file corresponding to a received log file, since the order in which the command files arrive to the receiver-SMTP is not necessarily the same as the order in which the corresponding logs come back. The solution to this problem is just to include an identifier to the command and log files, by adding a new command to those already existing in SMTP. This new command (TICK) will specify a unique number enabling a sender-SMTP to associate a log file to its corresponding command file.

Another drawback of this system is the need for the sender-SMTP to remember all the command files currently pending, to be able to match them with their corresponding log files when these come back. To solve this problem, yet another command (VERBOSE) is added to the initial set of SMTP commands, specifying that the log file will have to contain not only the responses to the command lines, but the latter as well. So, if the sender-SMTP trusts the receiver-SMTP, it need not keep a copy of the command file it sent since all the lines contained in it will be retrieved in the returned log file.

2.2.3 UUCP

On UUCP, USENET, EUNET and some other UUCP-like networks, mail is transferred between two adjacent hosts by using the UUCP protocol, which in fact is the usual way any kind of data are transferred on these networks. (In fact, sometimes other transport protocols are used, either in conjunction or in replacement of UUCP, as X.25 for example.) In addition to the transfer of data, this mechanism allows commands to be executed remotely. One of these commands is rmail (remote mail), which instructs a remote host to execute a mail command. But first, some details on how the UUCP transport mechanism works.

The basic operation of the network is very simple [Nowitz78]. The machines are connected by dial-up links (or by X.25 lines), which is quite cost effective and flexible. Each participating system has a spool directory, in which information on work to be done (files to be moved, or commands to be executed remotely) is stored. A standard program, uucico, performs all transfers. This program starts by identifying a particular communication channel to a remote system with which it will hold a conversation. Uucico then selects a device and establishes the connection, logs onto the remote machine and starts the uucico program on the remote machine. Once two of these programs are connected, they first agree on a line protocol, and then start exchanging work. Each program in turn, beginning with the calling one, transmits everything it wants to, and then asks the other one what it wants done. Eventually neither has any more work to be done, and both exit.

In this way, all services are in principle available between two connected sites. Furthermore, each caller knows the hours when each destination system should be called. If a destination is unavailable, the data intended for it remain in the spool directory until the destination machine can be reached.

The user has two commands which set up communications, uucp to set up file copying, and uux to set up remote command execution in case some of the required resources (system and/or files) are not on the local machine. Each of these commands will put work and data files into the spool directory for execution by UUCP daemons.

Now that we see a little better what the remote execution mechanism is, let's see how mail is transferred using it, with the help of an example [Horton86]. Even though source routing is increasingly discouraged on UUCP-like networks, it will be used here for a better illustration of the process.

Suppose we want to send mail to host1!host2!user, which means that the user is on host2 which is accessible from here via host1 (see section 2.4.3 for details on the UUCP addressing scheme). We type the command

```
mail host1!host2!user
```

The user interface program (mail) creates a file file1 such as

```
Date: 12 Apr 1988 11:22:53 GMT
From: ourname
To: host1!host2!user
Subject: test

This is the message.
```

and then passes it to the transport mechanism with a command like as

```
sendmail host1!host2!user < file1
```

More on the routing facility sendmail can be found in section 6.5.1. Sendmail prepends a From line, puts the result in file2, and passes this to uux with the command

```
uux host1!rmail host2!user < file2
```

This results in the command

```
rmail host2!user
```

running on host1. Seeing that the recipient is not local, rmail passes the mail to sendmail on host1, which prepends another From line to the message and passes the mail along by issuing

```
uux host2!rmail user < file3
```

where file3 contains

```
From nuucp Apr 12 12:43:35 1988 remote from host1
>From ourname Apr 12 11:21:48 1988 remote from ourhost
Date: 12 Apr 1988 11:22:53 GMT
From: ourhost!ourname
To: host1!host2!user
Subject: test

This is the message.
```

The command

```
rmail user
```

is run on host2. Then, since the recipient is local, rmail collapses all the From lines in only one From line, posting the message

```
From host1!ourhost!ourname Apr 12 12:43:35 1988
Date: 12 Apr 1988 11:22:53 GMT
From: host1!ourhost!ourname
To: host1!host2!user
Subject: test

This is the message.
```

in the mailbox of the addressee.

As can be seen from all this, the envelope of an E-mail message transferred via UUCP can be considered to be on the one hand, the arguments to the command rmail (for the destination) and on the other hand, the line which should start the standard input of the command rmail and having a format like "From host!user date remote from system" (for the origin).

It is possible that there will be additional From lines, each one of them having been added by each system the message passes through (as in the example). These lines are normally folded into a single From line on the destination system. However, it is also possible to preserve the forwarding date and system in a newly generated header line, such as Received or Sent-By.

Still other treatment of headers are possible, in particular when gateways are crossed. All this can be done thanks to the flexibility of mail routers like sendmail.

2.2.4 P1

P1 is the store-and-forward transfer protocol specified in the X.400 recommendations. It was already introduced in section 1.2 since it is the protocol used by the X.400 MTS to relay messages from the MTA of the origin UA to the MTA of the destination UA.

Execution of P1 between successive pairs of MTA's governs the store-and-forward transfer of messages throughout the MTS, and also specifies the way in which other MTL services are carried out, such as the generation and return of notices, the handling of probes, etc.

MTA's transfer messages and provide other MTL services by the exchange of Message Protocol Data Units (MPDU's). There exists two kinds of MPDU's.

User MPDU's (UMPDU's) carry user messages and control information. The message part contains the message that was submitted by a UA for transfer through the MTL and the control information part contains such indications as the name of the recipient(s), the transfer priority, etc. Normally, the message part is passed transparently through the MTL, except when conversion has to occur, for example when the format of the message is incompatible with the capacities of the destination UA.

The second kind of MPDU is the Service MPDU (SMPDU), which can contain either a probe or a positive or negative delivery report.

To implement the P1 protocol, an MTA is composed of three distinct functional entities (see fig 2.3). First, the Message Dispatcher constitutes the intelligence of the MTA. It performs all required processing operations on each MPDU passing through the MTA. Second, the Association Manager governs the establishment and release of associations (active communication paths) with adjacent MTA's. Third, the Reliable Transfer Server (RTS) supports the Association Manager and the Message Dispatcher by establishing and releasing associations, and by physically transferring MPDU's, respectively.

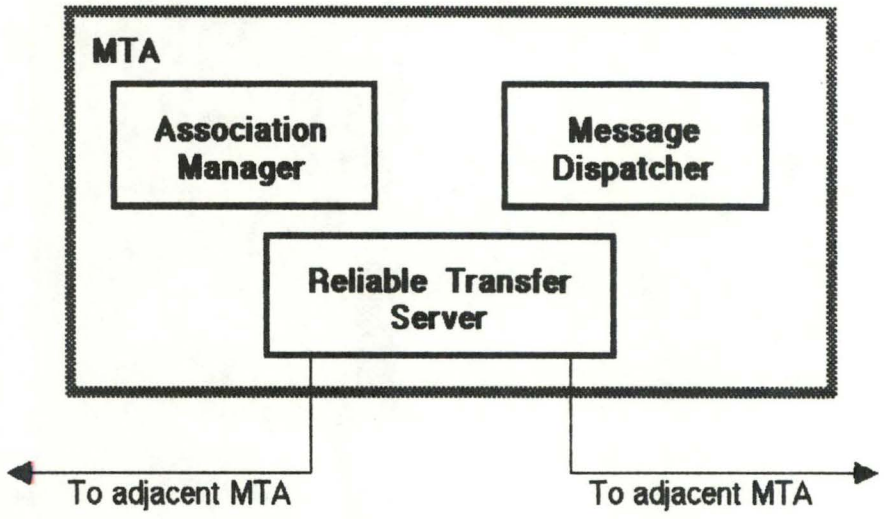


Fig 2.3 : Functional model of an MTA

Use of the RTS provides the Message Dispatcher with a simple, one step physical transfer capability. In fact, the RTS actually accomplishes transfer through complex interaction with a counterpart RTS via the OSI Session Service. Thus, the major function of the RTS is to provide a clear separation between the logical processing operations of the Message Dispatcher, on the one hand, and the complexities of physical data transfer, on the other. [Myer84]

As in SMTP, the transfer mechanism is quite efficient. Indeed, it is possible to specify several destinations for the same message and multiple copies of the MPDU containing the messages are created only when necessary, i.e. when the routes to the different destinations part from each other.

2.2.5 DECNET

Mail on DECNET is transferred using the mechanisms put at disposal by the DECnet underlying architecture, which is connection-oriented. This implies that a direct logical connection between the origin and the destination node must be established before any data can be transferred. On the contrary, on a store-and-forward system one just has to connect to the next host on the route to start the actual transfer.

DECnet supports adaptative routing, which permits data to be routed through the network over the most cost-effective path. Messages are rerouted automatically if a circuit becomes disabled.

To actually transfer mail between two different machines, the mail process on the origin machine asks to the DECnet underlying layers to put itself in communication with the MAIL object on the destination node. Thus, the mail process will take the rôle of the client, while the MAIL object will act as the server.

When the communication is established (no matter how, this is DECnet's problem), the mail process sends the different constituents of the message (to, from, the message itself,...) to the MAIL object, following a particular protocol. The MAIL object will then add the message to the recipient's mailbox and warn the latter that some mail arrived.

As will be seen in section 6.7.4, a special protocol (the Foreign Mail Protocol) allows access to mail gateways.

2.3 Overview of the major content protocols

Content protocols are those applying only to the format and semantics of the content of a message, and have normally nothing to do with the envelope of a message, and thus with the way a message is transmitted and delivered from its origin to its destination.

However, some message systems may use the information from the content to create the envelope and some content protocols have even been designed to facilate the acquisition of such information by programs (e.g. RFC 822).

When talking about content protocols, it is very important to remember that they determine the amount of functionality offered by the E-mail system considered. Not all content protocols offer the same level of functionality, which explains that some features of a mail system could be lost altogether when a message has to pass from an E-mail system to another, through an E-mail gateway. Indeed, when transferring messages between two systems using different content protocols, it is only possible to keep the functionalities common to both of them.

2.3.1 RFC 822

RFC 822 (Request For Comments number 822) is the standard protocol for ARPA Internet text messages. No special provision has been made for encoding drawings, facsimile, speech, etc. Full specification of RFC 822 can be found in [Crocker82].

A message conforming to RFC 822 consists of lines of text. It is composed of a heading and a body, separated by a blank line (see example in fig 2.4). The heading comprises rigidly formatted header fields, some of which are necessary in each message, some of which are optional. There is also the possibility to extend the set of standard fields with extension-fields in the future, as the need arises, and to include user-defined header fields, used privately.

It is interesting to note that header fields are composed of plain ASCII text, not binary codes as in P2 (described in the following section). This really means that one can build an RFC 822 message, complete with headers and body text, simply with a text editor and that such a message is easily processed by most text-oriented tools, including sophisticated mail router programs like sendmail.

Header fields are further structured in a field-name, then a colon (":"), then a field-body, and are terminated by a carriage-return/linefeed.

The most usual field-names are

- FROM, to indicate the origin user of the message
- TO, to indicate the addressee(s) of the message
- CC, to indicate secondary recipients (carbon or courtesy copy)
- DATE, to indicate the date of creation of the message
- SUBJECT, to indicate the subject of the message

Some of the field-bodies are structured, meaning they must be interpreted according to an internal syntax (e.g. FROM, TO, DATE,...) while others are said unstructured, because they are composed of a simple character string (e.g. SUBJECT, COMMENTS,...).

```
Date:     26 Aug 76 1429 EDT
From:     userA@hostA.ARPA
Subject:  example message
To:       userB@hostB.EDU
Cc:       userC@hostC.COM

This is an example message.
Bla bla bla
...
This is the end of the message.
```

Fig. 2.4 : Example of RFC 822 message

The field-bodies containing an address (e.g. FROM, TO, CC, ...) comprises a domain-dependent string and a domain reference, separated by a "@" character. The domain reference specifies a sequence of sub-domains, separated by dots. The domain-dependent string is uninterpreted, except by the final sub-domain (i.e. the leftmost one). The rest of the mail service merely transmits it as a literal string.

In addition to the functionalities already mentioned (indication of origin, destination(s), carbon copies, subject, date, comments), other offered services include indication of auto-forwarded messages, authorizing user and user who should receive a reply, blind copies (i.e. destinations of a copy which are not to be transmitted with the message), replies to a given message and encryption.

RFC 822 also provides an audit trail of the handling of each message by specifying that each host relaying a message has to add trace information at the beginning of the message itself, indicating the precise route followed by the message. This information can be used while trying to understand what happened to a given message or to automatically send a reply back to the origin of the message.

2.3.2 P2

P2 is the content protocol designed in the X.400 recommendations framework for the InterPersonal Messaging System. It was briefly described in section 1.3.

P2 offers all the functionalities provided by RFC 822, but is a richer protocol than the latter. Additional functionalities include the possibility to request a confirmation of the delivery of a message and the availability of other types of message than simple text (e.g. facsimile, graphics and in the future videotex and voice, plus any combination of these). It is also possible to indicate a date before which a message should not be delivered or a date after which the message is not valuable any more. One can also specify the urgency of a message as well as its sensitivity.

2.4 Overview of the major addressing schemes

When talking about addressing schemes, different terms have to be taken into account and distinguished. Firstly, a name is what is used to logically specify a resource (e.g. a mailbox, a host,...) on the network. Secondly, an address is what is used to physically

locate a given object. And finally, a route is the specification of the physical path joining two objects of the network.

Two main techniques exist to map host names to host addresses. The first one is to maintain on each host a map of all the hosts on the network with their corresponding address. This is bulky, slow to update, difficult to keep consistent but simple to implement. A more recent technique is the use, via special protocols, of nameservers which can be remotely located. These servers usually maintain only the information concerning that part of the network they are responsible of, and are able to get information about other parts of the network, via other nameservers, or at least are able to indicate the nameserver keeping the desired information.

There are also two major techniques at the disposal of users to specify the route to be taken. The first one is called source routing, which means that a user has to specify precisely all the successive intermediate hosts needed to reach the final destination. This is very annoying for the user. A second technique is not to specify any route at all, leaving the burden of finding the appropriate route to the underlying system. This technique allows a user to only specify the final destination, leaving to the system software the task of finding the actual route to take (e.g. by using network maps or nameservers).

Yet another routing technique is used when transferring mail between different E-mail systems. Then, a hybrid form of routing may be necessary and source routes like "alpha!beta%gamma@delta" are not uncommon. In addition to the fact that it seems rather cryptic, such an address may sometimes be ambiguous, i.e. interpreted differently depending on where it is interpreted (see section 2.4.3 for more details).

2.4.1 ARPA Internet

The ARPA Internet is organized in a hierarchical domain name system, which means that to specify the addressee of a message, one has to indicate the name under which he is known on his local system, then the list of subdomain(s) he is part of (the first of which being usually the name of the host he is on) and finally the name of his top level domain. The RFC 822 syntax is used. The format used is thus

user@host.subdom1...subdomn.domain

Top level domains include COM (for commercial organizations), EDU (for educational organizations), GOV (for civilian government organizations), MIL (for the Department of

Defence), NET (for administrative organizations for networks such as UUCP and BITNET) and ORG (for other organizations).

The use of nameservers allows to decentralize the administration of the mapping of host names to host addresses. Each nameserver controls part of the name space.

To fix our ideas, suppose that we want to send a message to user JAMES on a given ARPA Internet host [Quaterman86], whose name is SALLY.UTEXAS.EDU. The address of our recipient is thus JAMES@SALLY.UTEXAS.EDU. The name of the host means that it is locally called SALLY and is part of the UTEXAS subdomain (University of Texas), itself part of the EDU top level domain. The Internet address of this host could be something like 10.2.0.62. When needed by the mail program, this address would be requested from the system software of our machine, which would itself have asked a nameserver for it. The Internet protocol would then use the address to route our message to the appropriate network (in this case, 10 stands for the ARPANET). The communication subnet of that network would then use the rest of the address to determine a route to the destination host, using the local routing mechanism.

2.4.2 BITNET and EARN

There is currently no hierarchical naming of the hosts on BITNET. Each host is simply designated by its name. The network is organized in a tree structure, so that the route between any two host is unique. Regularly, an up-to-date map containing the name of all the hosts of the network (including EARN and NETNORTH) and the routes between hosts, is distributed to all the hosts. Thanks to this, users need only specify the name and the host of the person they want to send a message to, without mentioning any route.

For the future, North American representatives of BITNET have decided, in collaboration with their counterparts of the ARPA Internet and UUCP, to adopt the ARPA Internet domain naming syntax. On the other hand, EARN is planning to migrate to X.400.

This is one of the reasons why there are currently several methods for specifying addressees on BITNET. For example, the simple IBM mail program NOTE uses the format

user AT host or user AT host.domain

The MIT/Rice Mail program can use the same address formats but can also use RFC 822 to specify the address, thus

user@host or user@host.domain

In both cases, the second address format is used if the addressee is on another network.

2.4.3 UUCP and EUNET

UUCP is one of the few major networks still using source routing. Addressees are designated using the so-called "bang path", specifying each hosts on the route from the origin host to the destination host and separating them by exclamation marks ("bang" characters). The format of such an address is thus

host1!host2!...!host!user

which is not really easy to use and to remember.

This problem is somewhat reduced by using the UUCP map broadcast regularly by the USENET news facility to build (using the pathalias program) a database containing the routes between any two registered hosts. The mail system can then use that database to automatically compute routes, allowing users to simply specify "host!user" as destination of a message. There is of course a drawback, namely the fact that computing the database and storing it requires lots of CPU time and lots of disk space. Moreover, the database can never be up-to-date, due to time intervals between new releases of the map.

It is also possible for the user to send all non local mail to a router node (like the backbone hosts on EUNET) which will find the route to the final destination. A recipient address would then look like

routerhost!desthost!user

The advantage is that now only the router node has to know the paths to all the other hosts, for example by using a database such as the one described in the previous paragraph.

An additional problem comes from the fact that host names are not unique throughout the network. A single name could be assigned by several different companies to

several different machines. This may happen because a company was not connected to the general UUCP network at the time and thus was unaware of the conflict, or because a host was not originally expected to communicate with the world at large, or because the first machine having the common name was not listed in the UUCP map, or for other reasons [Quaterman86]. One way to solve this problem is to come back to the old-style bang path, which specifies a host unambiguously.

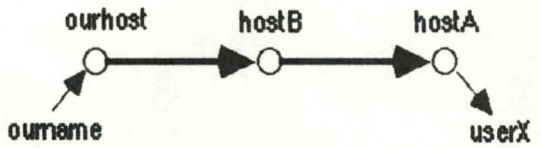
Another solution is under way to solve the addressing problem altogether. It was decided to implement naming in the style of ARPA Internet domains on UUCP and to discourage bang style addressing. This might also allow integration of the UUCP name space into the ARPA Internet domain name space.

However, the adoption of this new addressing format has given birth to so-called "hybrid" addresses, i.e addresses where exclamation marks ("!") appear to the left of a "@" sign, such as

hostA!userX@hostB

This address is ambiguous, since it could be interpreted differently. On the one hand, it may be seen as

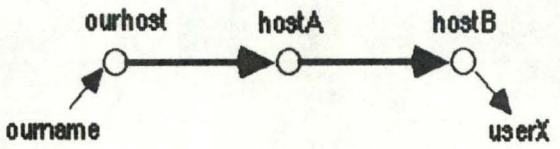
(hostA!userX)@hostB



which is the normal way to interpret an RFC 822 address, since such an address is considered to be composed of a domain (or host) specification (at the right of the "@" sign) and of a local part which is freely interpreted by the destination host. In that case, a message would first be sent to hostB, then to userX on hostA. This interpretation is the one in the hosts implementing the latest software and considering that UUCP/EUNET uses RFC 822 syntax rather than the old-style bang path.

On the other hand, the address could be interpreted as

hostA!(userX@hostB)



which is quite normal with respect to the original UUCP address format. In that case, the string appearing to the left of the first "!" is considered to be the name of the next host on the path to the destination host, and the rest of the address is uninterpreted. In this case, the message is first sent to hostA, and the remaining part (i.e. userX@hostB) is then interpreted (hopefully, hostA understands the RFC 822 format). This kind of interpretation is the one of hosts on the network which haven't yet adopted the new RFC 822 addressing scheme and which still uses the original bang path source routing.

EUNET has already almost completely eliminated the bang style syntax in favor of the

user@host

address format. Routing is managed by the national backbone hosts, each of which knows the organization within its own country and which hosts are in which countries. Moreover, the ARPA Internet domain naming syntax is currently being implemented on EUNET. Each country will register as a top-level country domain with the Internet (for example, BE for Belgium). This will simplify routing further since each backbone host need then only know the hosts within its own country domain and a path to the backbone host for each other country domain. There is no need to know anything about the internal structure of other country domains any more. [Quaterman86]

2.4.4 X.400

The X.400 addressing scheme has already been described in section 1.6. However, it is worth mentioning that existing X.400 implementations sometimes took some liberties with respect to the X.400 recommendations, in particular for what concerns the addressing scheme.

For example, the first version of EAN (EAN-1) used RFC 822 addresses, specifying a user, a subdomain list and a top level domain, instead of using the X.400 recommended addressing scheme where an addressee is designated by specifying attributes such as surname, organization, organizational unit, management domain etc. RFC was used to display and enter addresses. These addresses are however coded internally using the X.400 binary structures, using the mapping

user@domlist.domain -> /DDA1=user/DDA2=domlist/PRMD=domain

where DDA indicates Domain Defined Attributes. In the second version of EAN (EAN-2), , the RFC 822 addressing scheme is still used, thus still using Domain Defined Attributes to code addresses. However, EAN-2 is able to generate and display all kinds of X.400 addresses as well, i.e addresses of the form

/S=surname/OU=orgunit/O=org/PRMD=prmd/ADMD=admd/C=country

where S is the surname, OU is the organizational unit, O is the organization, PRMD is the PRivate Management Domain, ADMD is the ADministration Management Domain and C is the country.

This has the consequence that EAN-2 can talk with EAN-1 without problem but also that even though EAN-2 is able to send messages to real X.400 systems, these X.400 systems can send messages back only if they are able to generate addresses using the Domain Defined Attributes used by EAN-1 and EAN-2.

As for other X.400 systems, here are some of the existing implementations. The DFN version of EAN implements the recommended X.400 addressing scheme. It is also able to convert X.400 ORnames to talk to EAN-1 as well as to EAN-2. KOMEX is also a real X.400 system. GIPSI, the French X.400 implementation, is a real X.400 system too, but accepts RFC 822 addresses and is able to work as a gateway between these two worlds as well.

2.4.5 DECNET

The addressing scheme used on DECnet is not a hierarchical one as on the ARPA Internet, or a one which requires source routing as on the original UUCP. It is more like the EARN/BITNET one since all that is required to specify the address of a remote user is to quote the name of the remote host and the name of the user, separated by two colons. The address format is thus

node::user

This kind of addressing is transparent to the user who need not know all the intermediate hosts to a given machine. In fact, there are two kinds of node on DECnet.

The first category is called "end nodes". The second one is composed of "router nodes". The router nodes know, thanks to dynamic local tables, the different possible routes to follow to establish a logical connection to any node of the network. End

nodes have to pass via router nodes to establish a communication path to remote hosts.

Chapter 3
The functions of an E-mail gateway

Generally speaking, gateways are used to connect computer networks. A gateway is a computer system which switches data between networks, provides routing information across networks, and performs necessary format and protocol translation [Redell83]. The complexity of a gateway depends on the similarity of protocols used on the networks being connected. It also depends on the level at which the gateway will operate.

When talking about E-mail gateways, these generalities remain true. But more specifically, a mail gateway can be considered as a computer that is connected to two or more networks and which relays between them messages from the different E-mail systems. For this to happen properly, provision has to be made to ensure a correct mapping of addresses and headers, to provide body conversion when necessary, to generate error messages and log data as appropriate.

But before we have a closer look at all these functions, two major techniques used to implement E-mail gateways are presented.

3.1 E-mail gatewaying techniques

The first technique described hereafter to implement an E-mail gateway implies modifications at the Message Transfer Layer level while the second one has only some impact at the User Agent Layer level. Respective advantages and inconveniences follow from this distinction.

3.1.1 Relaying approach

In the relaying approach, a message intended for a user on another E-mail system is first routed by the MTS of the original E-mail system to a machine that, in addition to being part of the E-mail system, has the special function of being one of the entry/exit points of it (see fig 3.1). Such a machine is sometimes called an Entry/Relay Node (ERN). There is a corresponding ERN on the other E-mail system. Between these two

ERN's, there is a gateway, which is most of the time hosted in one of the ERN's. It is also possible that a particular host is part of both E-mail systems. In that case, all functional units are combined in the same host.

The forwarding of a message to the gateway happens either because the ERN was explicitly named in the address of the recipient, or because the local E-mail system was smart enough to realize that the destination host was not on the same network. In that case, it has somehow determined the name of the appropriate ERN (for example, by looking in static tables or by calling the services of a nameserver).

In the relay approach, the gateway is directly linked to the ERN's, which are part of the E-mail systems, and can thus be considered to be at the intersection of these E-mail systems, at the MTS level.

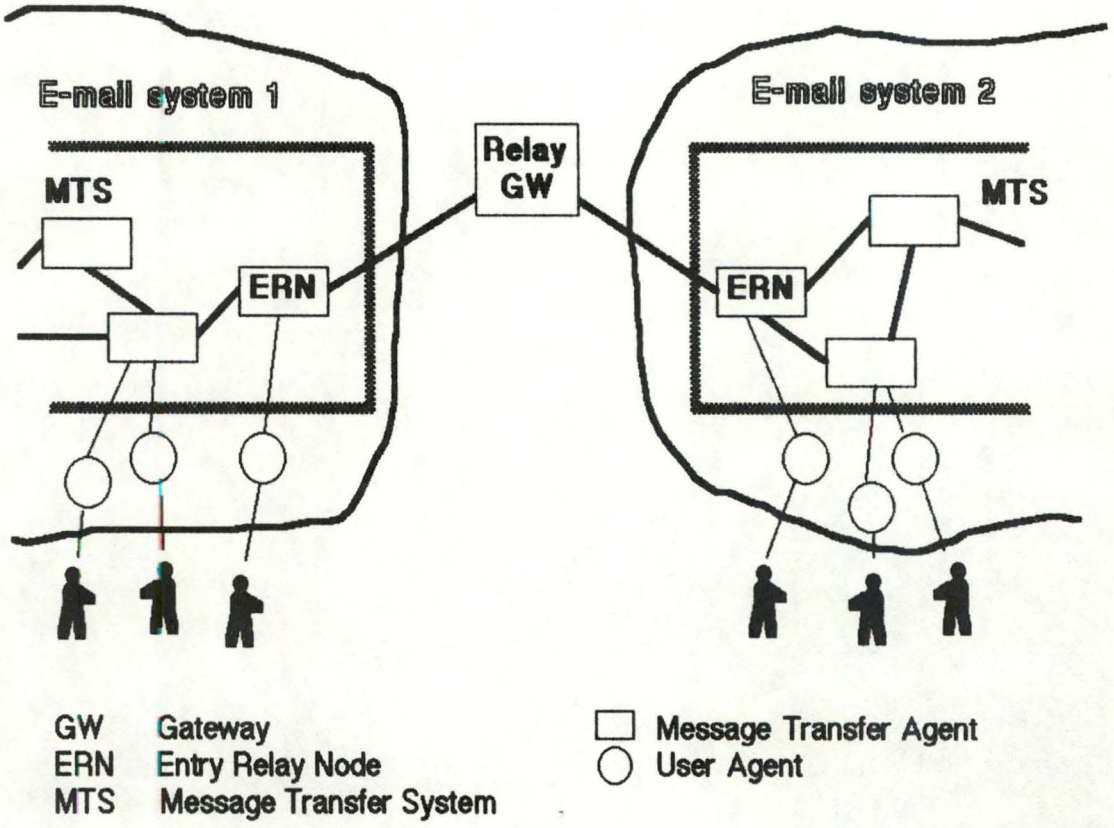


Fig 3.1 : Relaying approach

The gateway software transforms any message coming from the first ERN according to the destination network standards, and forwards the message via the second ERN, on the appropriate E-mail system using the protocols in use there.

In this approach, it is the message transfer systems of the networks involved that are responsible of the proper transfer of the message from one network to the other. Thus, implementing a mail gateway using this technique implies some modification at the MTS level in the networks interconnected, at least in the ERN's.

3.1.2 User Agent gateway

An alternative to the relaying technique is to create on each of the E-mail systems that are to be interconnected a dummy User Agent representing the entry/exit point to other E-mail systems (see fig 3.2). When someone on the local system wants to send a message to a user on another E-mail system interconnected with the UA gateway technique, he has to specify as recipient the name of the gateway UA itself. The actual recipient of the message has then to be mentioned somehow, for example with a line of the form

```
>>user@host(options)
```

in the message itself.

The gateway is implemented by a process whose job is to have a look at the mailboxes maintained by all the artificial UA's, to collect the messages that it finds and to post them on the appropriate E-mail systems, still using the artificial UA's as intermediary.

In this approach, there is no direct intersection between the interconnected E-mail systems.

Two packages implementing such a UA gateway are MLNET and a part of COSAC. Here follow a few details on how they actually work.

Regularly, for example once a day, the process implementing the gateway wakes up. For each of the E-mail systems interconnected, the following happens. Via terminal emulation, that process pretends to be a normal user associated with the artificial UA. Then, it reads the messages it finds in the mailbox maintained by that UA.

Each message is then treated, i.e. the special line containing control informations (actual recipients, options,...) is extracted and the message itself is put in a standard format for the gateway (i.e. RFC 822 for MLNET and X.400 for COSAC).

The format of the message is then changed to the one of the recipient's E-mail system and the message is posted to that system, using the same principle as for the reading of the messages.

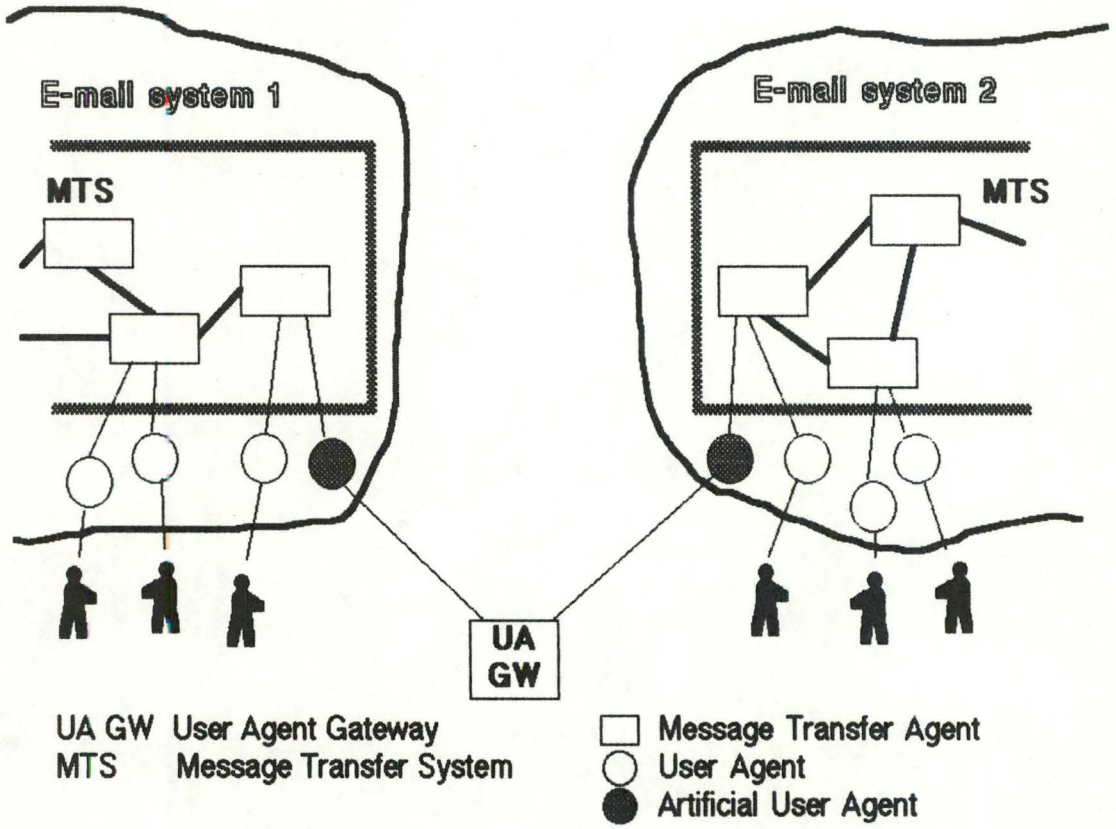


Fig 3.2 : User Agent gateway technique

The major advantage of this technique is to allow the interconnected E-mail systems to take a passive role, and to ignore altogether the existence of the gateway. The corresponding drawback is of course that someone has to know that there is a gateway somewhere, and if it is not the E-mail systems, it must be the user. The latter himself has to address his messages to the special gateway UA and also has to code control information in the message body itself.

This approach also appears as being simpler, since no modification is done to the existing E-mail systems. For example, a new type of E-mail system can be interfaced to MLNET rather easily by specifying 4-5 pages of a script that describes the dialogue for logging into the system and reading and writing the messages. And no more than about 500 lines of C code have to be written for the message format manipulations in both directions. [Beyschlag85a]

However, the apparent simplicity of this technique also implies substantial loss of functionality and other problems (see sections 4.1 and 4.2). In addition, the delivery delay is normally higher since the gateway is not active all the time, as in the relaying approach.

3.2 Mapping addresses

A gateway should map addresses in all the header fields of a message so that they conform to the formats used in the destination mail system [Heagerty87a]. This is necessary if functions implying some kind of reply are to be provided.

For example, it is necessary if an E-mail system allows its users to automatically reply to a given message, without the need for him to retype the correct address. It is also necessary when an error message has to trace its way back to where the original message came from, as well as for a delivery or reception notification to be properly sent back to whom it is intended.

This mapping of addresses also means that users see addresses in the form they are used to and which they would enter themselves.

The way mapping between addresses is to be realized must be specified somehow and implementors of E-mail gateways have to conform to these specifications to ensure consistency. This is especially true for what concerns the two most important addressing standards at present, i.e. X.400 and RFC 822. X.400 is important because this standard is the one towards which all E-mail systems are migrating and will eventually use. RFC 822 is equally important, because it is at present the de facto standard, and is used on several major networks including Arpa Internet, EARN/BITNET, EUNET/UUCP and CSNET. JANET uses a mail protocol (known as Grey Book) which is also based on RFC 822.

Two proposals currently exist to specify the mapping between X.400 and RFC 822 addresses. RFC 987 is the proposal coming from the Arpa Internet [Kill86] while there is also a proposal coming from the German research network DFN [Henken87].

3.3 Mapping headers

The level of functionality offered by an E-mail system is highly dependent on the variety of headers that can be specified. Because of this, the level of functionality a mail gateway can maintain is also highly dependent on the mapping it can realize between header fields.

Of course, it is not always possible to find a perfect mapping of headers, for example when the services offered in the interconnected E-mail systems are different. In these cases, some loss of functionality is inevitable (see section 4.1) and the gateway cannot be transparent to the users any more.

A possible long-term solution to this protocol conversion problem is for all interconnected E-mail systems to migrate towards a common functional specification [Redell83]. This solution defines a series of services as being standard and all E-mail systems are asked to provide functionally equivalent services. The mapping of headers will then always be possible for these standard services, without loss of functionality. The services which are too sophisticated or to specific to be requested from all E-mail systems are considered optional and it must be possible to determine if a given E-mail system supports them or not.

An even better solution is the migration to an E-mail standard, since in this case, the objective is not to ease the job of mail gateways, but rather to suppress the need for gateways altogether. X.400 is a major step in this direction.

But in the short term, some rules have to be fixed to guarantee a mapping which is at the same time optimal and not too complex, and being respected by everyone. Mappings should of course not require any changes to end systems.

An example of mapping "standard" is the RFC 987 recommendation which specifies the mapping between X.400 and RFC 822, at the level of service and protocol elements, character sets and, as seen in the previous section, addresses.

3.4 Converting bodies

The previous section addressed the problem of the conversion of headers, which ensure the smallest loss of functionality when a message has to cross a gateway. Another important and related problem is how to ensure that the body of the message is passed properly.

In fact, the problem of format conversion is twofold. On the one hand, there is the issue of representation conversion. This can occur for example when a message comes from a network where the standard text encoding conforms to EBCDIC and is bound to a network where ASCII is used.

On the other hand, there is the issue of medium conversion. Some E-mail systems support body formats ranging from simple text to voice, including facsimile, graphics and teletex while most systems are only able to process plain text messages. Once again, some conversion standard is needed to ensure consistency.

Incidentally, this problem is not specifically linked to gateways. It also exist in E-mail systems where several body types exist but where not all the User Agents are of the same degree of sophistication. For example, on X.400 systems, the MTS is responsible for the conversion of messages which are intended for UA's which do not support the type of message body that should be delivered.

To solve the format conversion problem, two approaches are possible [Redell83]. The first solution requests a gateway to be able to convert from any source format to any destination format existing on the networks connected. This approach quickly becomes unmanageable for two main reasons.

Firstly, the number of pair-wise conversions increases as the square of the number of formats. Secondly, some formats are proprietary and thus the knowledge required to convert to or from them may not be public domain.

The second approach is to define a standard format and to provide in a gateway only those facilities needed to convert between the local formats and the standard format. Such a standard format is called a document interchange format (DIF).

To define the DIF, there are again two approaches possible. In the first one, the DIF represents the lowest common denominator of all formats, so that all formats can be derived from it without loss of information or functionality. This ensures consistency

between the networks but prevents the more sophisticated formats from being used effectively, since the information that makes them sophisticated is lost when they are converted to the DIF.

On the contrary, the second approach defines the DIF as being general enough to encompass all other formats, and making it possible to code in the DIF any information contained in the most sophisticated formats. This guarantees no loss of functionality for networks using these formats, but implies some inconsistency since the DIF will be converted differently to formats with less sophistication.

3.5 Reporting errors

A gateway must be able to generate error reports of two kinds. First, an error may occur in the gateway itself, for example if the gateway is unable to deliver a message because the destination host and/or network is unknown in its routing tables.

Second, a gateway must also be able to forward error messages from one E-mail system to another, for example when a message was undeliverable on the destination host/network because the specified user was unknown there.

Most of the time, error messages are cryptic, incomprehensible for the non-specialist user who will receive it. One of the reasons for this is that it is not always possible to match perfectly error messages between different E-mail systems. For example, the E-mail systems used over DECnet can generate a "remote node is unreachable" error indicating that a direct connection to the specified node was temporarily not possible. When that kind of message has to cross a gateway from an E-mail system running over such a connection-oriented network to a store-and-forward E-mail system like X.400, there is a problem of matching because the concept of end-to-end connection is unknown in a store-and-forward system. The task of the gateway will then be to try to explain the error as well as possible, in terms understandable by the user which will receive the error message.

Error messages may be forwarded using the error reporting mechanism provided by the E-mail system involved, if this is possible. For example, there are provisions in X.400 to indicate to a user that his message could not be delivered (non-delivery indication). However, sometimes the only solution for a gateway to convey errors is as an ordinary text message [Heagerty87a].

3.6 Logging trace information

Logging trace information during the activity of a gateway consists in writing somewhere (e.g. in a file, on a printer or on a console) information related to each of the messages processed. This information should at least contain the origin of the message, the destination of the message, its size, the time and date of processing and whether the message was successfully accepted by the E-mail system it was passed to [Heagerty87a].

The logging of this information is of absolute necessity for the gateway managers to be able to play their role properly. Section 5.3 describes in some more details how the logged information is used by gateway managers.

3.7 Interface to E-mail systems

The issue of interfacing a gateway to the E-mail systems it is supposed to interconnect is not directly linked to the functions the gateway is to execute, but can have some significant impact on how well the job is done and on the ease of implementation.

The goal of such an interface is to ensure that a message headed for a foreign mail system is passed properly to the gateway and similarly that a message is correctly inserted in the destination E-mail system.

The ease of implementing this interface depends on the particular E-mail systems, but more recent manufacturer's products have started to provide programming interfaces to their mail systems [Heagerty87a].

Indeed, most gateways are implemented using one of a limited set of products that were developed with that goal in mind. These products are sometimes called Internet Routers. The best known of them are sendmail (described in detail in section 6.5.1) and MMDF.

Since these programs are widely used and are not too numerous, the developers of E-mail systems increasingly provide programming interfaces to the Internet Routers used in the environment where their own products will have to operate. For example, mail programs running under Unix often include interfaces to sendmail, as is the case with

the Unix mail and the EAN package. These and other programming interfaces will be described in chapter 6, where a real gateway system implementation is presented.

It must also be noted that developers draw more advantages than inconveniences from providing these programming interfaces, since otherwise, their products would be more difficult to integrate with existing ones. Even if these products were better, they would have to take existing systems into account or take the risk of being disregarded altogether.

Chapter 4
Problems associated with E-mail gateways

At the beginning, there were several networks, each having a particular E-mail system allowing their own users to communicate. Then, users on different networks began to desire to communicate with each other. The solution was provided by the implementation of E-mail gateways. Now, anyone can send messages to anyone else in the world, using that worldwide internet constituted by all the isolated networks linked by gateways. So, all is perfect. But, is it really ?

Beside the major advantage of offering a much greater connectivity, mail gateways also have their drawbacks, linked to the use of different content protocols and addressing schemes. These problems will be detailed in this chapter, first stressing the user's point of view with functionality and addressing issues, then describing two typical problems of gateway managers, namely looping messages and name clashes.

4.1 Loss of functionality

The overall functionality of E-mail that passes through a gateway is related to the message contents and the various header fields. The greater the similarities of the end-to-end protocols , the better the overall functionality.

When no perfect mapping can be achieved between the header fields specified in the content protocols of the interconnected E-mail systems, some loss of functionality is inevitable. This may happen when services offered on the different systems are not the same. Then, the maximum level of service that can be maintained by the gateway corresponds to the lowest common denominator of services between the E-mail systems involved.

In addition to the inconvenience it brings, the fact that different sets of functionalities are offered by different E-mail systems can lead to some inconsistencies. For example, suppose the E-mail system EMS1 provides confirmation of message delivery, and that E-mail system EMS2 doesn't. User A on EMS1 sends a message to user B on EMS1 and to user C on EMS2 and requests confirmation for the delivery of the message (see fig 4.1). The service provided to him will be inconsistent, since he will receive

confirmation of delivery to B but not to C, the gateway being unable to encode the request for confirmation when passing the message on EMS2. [Redell83]

An alternative to this solution is for the gateway recognize the confirmation requests, realize that the destination E-mail system is unable to cope with such a request, and send itself a confirmation to the origin user, clearly indicating that the confirmation comes from the gateway and explaining why.

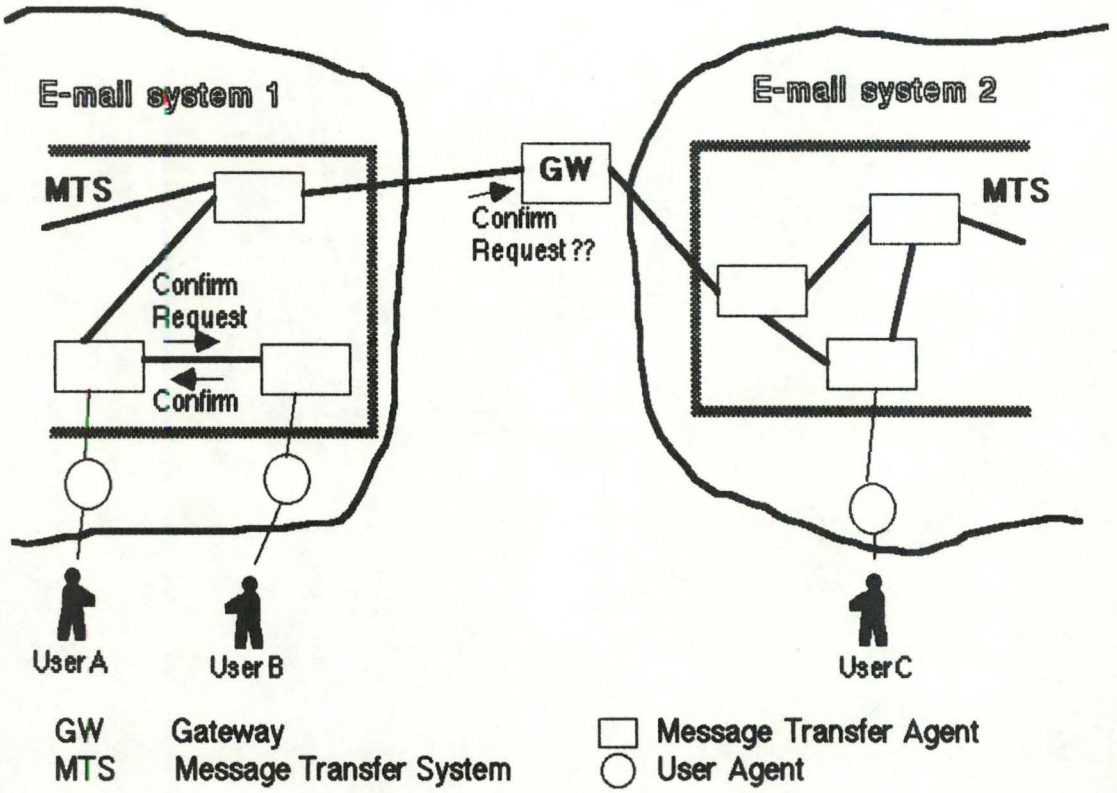


Fig 4.1 : Example of inconsistency of services

The User Agent gateway technique (see section 3.1.2) brings additional loss of functionality. For example, it is usually impossible to use distribution lists and aliasing facilities. Moreover, most of the time, recipients are prevented from replying. All these problems come from the fact that a message having crossed a UA gateway appears to have been submitted by the gateway User Agent itself, rather than by the original user.

Another kind of loss of functionality may be incurred when a gateway is crossed. Indeed, when a voice or videotex message will (in the future) cross a gateway to a simple text-only E-mail system, there will be an obvious loss of functionality.

4.2 Addressing

When addressing a user on the same network, one simply has to use the conventional addressing scheme of the network's mail system. This generally implies specifying the addressee's name and the name of the host on the network. But when one has to address someone on another network, gateways get involved, and the fact that messages will somehow have to pass through them is most often not transparent to the user.

Three addressing syntaxes can be distinguished, depending on the characteristics of the originator's network and of the gateway [Beyschlag85a].

4.2.1 Transparent addressing

With transparent addressing, all addresses consist of a recipient name, a host name and a network name (the network name might be omitted if the addressee is on the same network and the host name as well if the addressee is on the same host). Using the RFC 822 format, this would look like : "user@host.network". The E-mail systems themselves would take care of the routing of messages, using the appropriate gateways if necessary.

This solution is the most satisfactory for the user, since the passage through a gateway is completely transparent. This is even more so in cases where several gateways have to be crossed. Another important advantage of this technique is that any changes in gateway names or gateway locations are transparent to users and are taken into account automatically by the E-mail systems, with no risk of error.

The problem is of course that it is not so easy to arrange that each host on each different E-mail system is able to accept such a general address format. Furthermore, this would imply the use of a kind of directory service indicating which gateway to send a message to in order for it to be received by the destination user.

A lot of work remains to be done in this area, but the example of the ARPA Internet is worth considering since it is in fact composed of heterogeneous networks. Transparent addressing is used, but is surely eased by the fact that all the networks use the same mail protocols.

Once again, the problem of transparent addressing to other E-mail systems is only an extension of the same problem at the E-mail system level. There too, one would like to specify an addressee with just something like "user@host", without specifying the route that has to be taken.

4.2.2 Source routing

A second way to address a user on another E-mail system is to use source routing. In this case, an address has to mention explicitly the name of the gateway to be used to pass a message on another E-mail system. An RFC 822 address of this type can have two forms.

The first one is in fact used to specify source routing within the RFC 822 addressing scheme, i.e. with no special consideration for other E-mail systems. This kind of source routing is indicated by

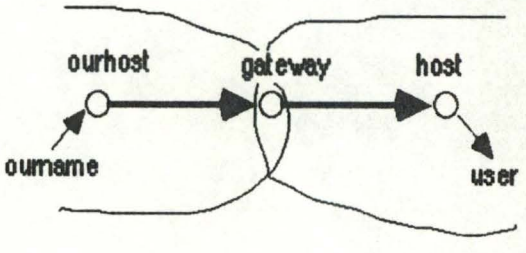
@gateway: user@host

but is discouraged.

The second way to use source routing with an RFC 822 address takes advantage of the fact that an RFC 822 address is in fact composed of a local part and of a domain specification (as seen in section 2.4.1), separated by an "@" character. The domain part must comply to the format specified by RFC 822, but the local part can be virtually any character string. Indeed, this local part, as its name indicates, is only interpreted at the final host, and is passed untouched by the intermediate relay hosts. So that some informal conventions have been born, specifying some rules to interpret local parts containing special characters.

For example, to sent a message on another E-mail network, the following RFC 822 address could be used :

user%host@gateway



The message would first be sent to the gateway host, as for any other message to any other host. Then, the gateway software would interpret the local part of the address, i.e. "user%host", replacing the last "%" by an "@", thus transforming it to something like

user@host

and would then decide where to forward the message next (incidentally, possibly to another gateway since the local part could also look like "user%host%gateway2").

It should be emphasized that this kind of source routing isn't official, but rather is purely conventional. The percent sign is commonly used to indicate source routing, but other special signs exist as well.

Another example of the use of such "hybrid" addresses can be found on the UUCP/EUNET network where, as has seen in section 2.4.3, two different addressing formats are used (old-style bang path and RFC 822 format), sometimes at the same time. This may result in addresses of the form

host1!local@gwhost

which can be interpreted differently if the UUCP point of view is used or if the address is considered to be in the RFC 822 syntax. In the first case, the message must first be sent to host1 and from then to the user represented by the local part "local" at the gateway "gwhost". In the second case, the message is sent to the gateway host "gwhost" and there, the local part, constituted of "host1!local" is interpreted.

The use of this type of source routing is more demanding for the user, since he has to know the precise name of at least one gateway to each of the networks he has addressees on. Furthermore, names of gateways are likely to change, and also it is not always obvious which route is optimal when several exist. But this format has the advantage that the user has more control on the actual route taken by his message. This can be important in the (not really frequent) case of highly sensitive messages which one does not want to pass through a less secure gateway.

Moreover, the fact that the formats of these addresses are not standard and may even be different for each of the E-mail systems interconnected makes this kind of addressing a little more confusing.

Another drawback with this solution is that it could prevent the recipient of a message to reply to his correspondent. This depends on the quality of the gateway software and the recipient's mail software.

4.2.3 User Agent gateway

The third possibility for addressing someone on another network is when the gateway is implemented by the User Agent gateway technique described in section 3.1.2. In this case, the user has to specify as destination the gateway UA itself, and encode somehow the actual recipient in the contents of the message.

This technique is clearly the worst for the user. It constrains him to use totally different addressing for a recipient on the same or on another network. In the latter case, it is not the address of the recipient that has to be specified, but the address of the gateway UA, which is at least a little disturbing and of course not transparent at all.

When this technique is used, replying to messages having crossed a gateway cannot usually be automatic (i.e. without the need for the user himself to specify where to send back a reply) since the message appears as having been sent by the gateway UA, not the original sender.

Moreover, when a problem involving a message coming from a UA gateway is detected by another gateway or E-mail system the message has passed through, the error notification is sent to the gateway UA, not to the one who sent it. Even if the UA gateway included additional code to treat error messages, this would be unusable, since by the time an error notification reaches the UA gateway, a reference to the original user is lost.

4.3 Looping messages

The problem of looping messages is well known in E-mail systems. It is also present for several reasons in E-mail gateways.

One of the causes having as consequence that a message is passed continually between mail systems and is never delivered to a user has to do with mailing lists and is described next [Heagerty87a].

Consider two E-mail systems EMS1 and EMS2. There is a distribution list on EMS1 and one of the members of this list is user X on EMS2. Suppose further that X has set his mail system to automatically reply to any message with another message like "I'm on holiday. I'll call you back". If a message is ever sent to the distribution list, it will be forwarded to user X, and the mail software of this will automatically generate a reply. Although when in its own environment the mailing list mechanism could recognize that message as an automatic reply and not redistribute it, once the reply has crossed the gateway, it appears as a normal message and is sent to everyone in the list. This in turn causes another reply, and this goes on until someone notices the problem and acts consequently.

Likewise, when a distribution list contains a faulty address, the resulting error message is often redistributed, also to the faulty address. This can cause congestion of whole networks.

Another type of never stopping message is called "bouncing mail". It can occur when each of two interconnected hosts believes delivery of a message should be via the other one. The consequences of such an erroneous behaviour can be kept at a tolerable level by limiting the number of "hops" a message is allowed to pass through.

4.4 Name clashes

Name clashes occur when an address corresponds to more than one destination. This happens for example when two or more hosts may have identical names, as is the case on UUCP, where there was originally no central administration.

Usually, name clashes are prevented thanks to the fact that on most networks, there is a central authority where all hosts and domains have to be registered. This ensures the uniqueness of names. Incidentally, UUCP is heading towards such a policy, the current anarchy making a proper management very difficult.

But when gateways get involved, name clash problems appear again, even if there is no such trouble on the interconnected networks when taken separately. They appear because there is no supra-network authority that would assure absolute name uniqueness, at least in each country. Inside the gateway, this may lead to the same mapping for different destinations.

For example, suppose that on three different networks, EARN, JANET and X.400, in country CC, there is a host called hostX and a user named Smith [Heagerty87a]. The corresponding local addresses are respectively Smith@hostX.CC (RFC 822), Smith@CC.hostX (JANET uses protocol Grey Book similar to RFC 822 for the addressing, except that domains are written the other way round) and /S=Smith/O=hostX/C=CC (X.400). Because the networks are not connected, there is no problem.

But suppose now that a gateway is set up between these three networks. After having crossed the gateway, all the addresses mentioned above will map into the same address, i.e. Smith@hostX.CC if RFC 822 is the common addressing syntax of the gateway (see fig 4.2). Then, there is no means of knowing to whom the message is to be sent.

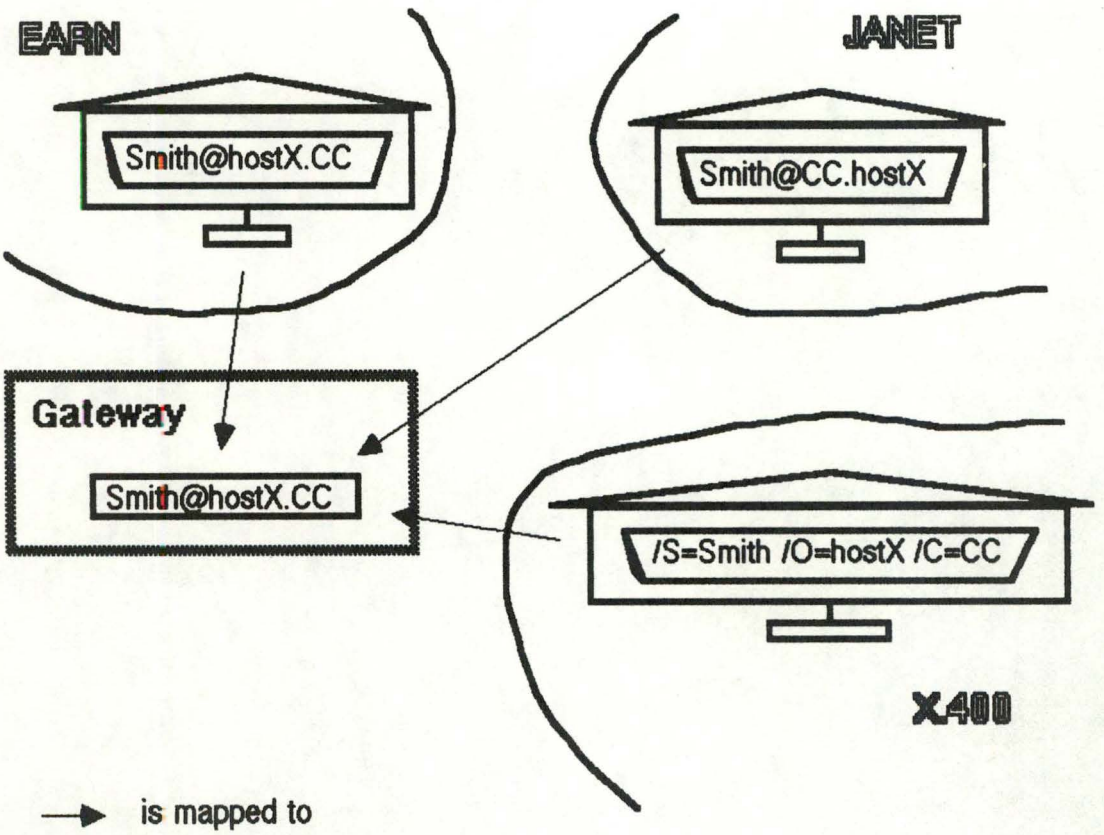


Fig 4.2 : Example of name clash

That is why, nowadays, most countries are arranging joint meetings between the different networks, so that the global namespace can be coordinated, thus avoiding

name clashes altogether. In fact, as there is no technical solution, it needs to be administrative. Different networks within a country must sooner or later interconnect and coordinate their naming. Incidentally, one of the aims of RARE is to push countries to organize themselves in this way.

Chapter 5
Management of E-mail gateways

The interconnection of several E-mail systems is not a trivial problem. But the operation and maintenance of mail gateways is not an easier task. E-mail gateways are fine mechanisms that have to be looked after very carefully.

Beside technical problems associated with the running of a gateway, other issues must be taken into account. They address accounting, administrative and legal problems.

The tasks of a gateway manager encompasses both kinds of problems and are described hereafter. This chapter is mainly based on several papers about the practical experience of gateway management. They were written by a gateway manager at CERN, and summarized in [Heagerty87a].

5.1 Updating routing tables

A gateway can be seen as a special host connected to two or more E-mail systems. When a new host or a new domain (for some E-mail systems) is added, some kind of updating has to be done, as is necessary for any ordinary host.

The updating can consist of the modification of some local static tables or can be taken over by a name server. For example, on EARN/BITNET, complete maps of all sites are regularly broadcast to all the nodes on the network, while on the Arpa Internet, only the tables of the name servers involved have to be modified.

But beside the tables peculiar to a given E-mail system, the gateway manager is responsible for additional tables specific to the gateway.

A case highlighting the necessity of these specific tables is described hereafter. Imagine a gateway interconnecting more than two E-mail systems. When receiving a message from one of the E-mail systems, the gateway has to know to which E-mail system to forward the message. For example, when a gateway interconnecting UUCP, BITNET and EAN receives from UUCP a message which was originally addressed to

"user%host@gateway", the gateway tables have to be consulted to find on which network to sent the message, BITNET , EAN or even UUCP.

Very often it does not suffice to simply update a table of hosts belonging to a specific network. This information must be incorparated in the appropriate way in the gateway programs, e.g. some programs may store host lists in a hash table, other programs have them compiled in for efficiency reasons.

5.2 Checking connections

A good means of being sure that there is no blocking, loss of connection or other problems is to automatically and regularly send test messages around the major routes and gateways. This ensures that any problem with the connections is quickly detected and resolved.

Moreover, gathering statistics related to these test messages allows a gateway manager to fine-tune his system. For example, mean trip times of test messages can indicate where bottlenecks are and in what circumstances congestions occur. It also enables the gateway manager to determine more suitable call frequencies to certain destinations.

5.3 Checking log files

Log files contain the information on which future decisions and priorities will be based. Checking the log files is an essential part of a gateway manager's tasks for the following reasons.

First, it can explain why some messages were treated improperly, allowing to trace the information back through several gateways if necessary. For example, the trace information can show that a message could not be treated properly because it was too large or because the number of recipients was too high.

Then, it provides the crude information from which statistics about the use of the gateway can be calculated. Likewise, when users are to be charged for the use of a

gateway, the accounting can be based on the logged information. Moreover, analyzing log files can help detect illegal traffic or other misuse of the gateway (see chapter 7 for more details on these topics).

Logged trace information can also outline user problems when they have to use gateways. Indeed, it is sometimes really baffling to see how exotic some addresses to foreign E-mail systems can look. It is even more surprising to realize the number of such extravagant formats some gateways can cope with.

When the number of refused messages is too high, for example because of bad address formats, gateway managers have to do one of two things : either they have to include in the gateway additional software or tables to enable correct parsing of the previously unrecognized address formats, or they have to edit a guide containing the correct address formats for messages that have to cross the gateway. Reference [Heargerty87c] is such a guide and was written by one of CERN's gateway managers, mainly for internal usage.

5.4 Analysis of statistics

The regular analysis of the statistics gathered during the activity of a gateway allows a gateway manager to check the proper functioning of the system he is responsible of, and to take appropriate decisions for the future.

First, the statistics produced automatically give precise figures concerning the actual traffic passing through the gateway, figures that are quite difficult to assess otherwise. This in turn helps the gateway manager to decide what resources he has to assign for the service to be provided properly.

The most popular hosts (i.e. those who receive/send most messages) can be easily detected and the mean traffic volume involving them can be estimated. From these figures, the gateway manager can decide what type of connection there should be to the different hosts as well as the frequency of connection with them. Would a leased line be more cost-effective than the current dial-up line to this host ? Shouldn't we establish a direct connection to that host, rather than to have to pass through relay hosts ?

It must be noted that this attitude is not in fact specific to gateway managers but rather is the normal way of thinking for each responsible of a local E-mail system. Indeed, he must also decide what type of line to use to connect to a neighbouring host on the same E-mail system. This is especially true for router hosts, or hosts being part of some kind of backbone in an E-mail system.

From the size of the messages bound to certain hosts, the question can also be asked whether it wouldn't be wiser to establish a file transfer service instead of or additionally to an E-mail service. Interpersonal messages are indeed usually not huge, and more appropriate services exist for other kinds of transfer.

The growth rate of the use of a given gateway, derived from the successive statistics, can also help a gateway manager to estimate when the system will be overloaded, or conversely if the gateway is less used than before.

Another very important information that the analysis of statistics can provide is the approximate cost of the gateway service, at least as far as the actual transfer of bytes on the links to and from other hosts is concerned. The calculation of this information must take into account several factors, including the nature of the links (e.g. dial-up or leased line), the cost per volume and per time on these lines, and still others.

The implementation of a particular tool to produce statistics from log files for an existing gateway is described in chapter 7.

5.5 Charging and accounting

Running a gateway implies the use of a substantial amount of resources, which have to be paid for by someone. Sometimes the gateway is operated for the benefit of a whole community and is funded centrally. This is more or less the case for CERN's gateway system which was originally set up for private use within the High Energy Physics (HEP) community. In other cases, users have to be charged some way for the use of the service offered.

Anyway, it is always interesting to know who uses the gateway and for how much, at least to encourage a cost-effective use of the service.

5.6 User support

As we have already seen, an E-mail gateway can usually not be totally transparent for users. When he has to send his messages through a gateway, a user is often confused, for several reasons. He must use a special addressing method, he has to be aware that the functionalities he is used to might be lost when a gateway is crossed, he has to understand new types of often cryptic error messages, etc.

To cite one of CERN's gateway managers, "As the technical problems of internetworking different X.400 products and gatewaying with existing mail systems are being overcome, the current challenge is to adopt X.400 systems with minimum confusion to the user community" [Heagerty87b]. This stresses the point that the problem is less technical than organizational, i.e. that the feasibility of gateways is proven, while the practical use of them by non-specialist users still causes problems.

To help users utilize gateways in a proper manner, gateway managers have to provide the necessary information. For example, they can provide, in collaboration with E-mail responsibles, user guides explaining the different address formats to reach someone on another E-mail system.

Some kind of on-line help is also appreciated. For example, some mail software (or more generally, some operating systems) allow system managers to add help screens to the help facilities already provided. So, a help screen containing address formats to the main destinations where a gateway is involved could be added to the standard on-line help.

Another kind of support provided to users on the same site as the gateway (and possibly for others too) is the direct answer to questions. For these users, a site-wide electronic mail directory service can also be setup, containing the names and addresses of all the users on the site. This is being done at CERN (see section 6.9).

Another way CERN's gateway managers have facilitated the life of users is by arranging so that all major E-mail systems on the site use a common address format, namely

user@host.domain

This common addressing scheme makes no reference to any gateway machine and leaves the job of correct routing to the gateway system, in conjunction with the E-mail software corresponding to the different E-mail systems. This E-mail software is supposed to send all non-local mail as mail for which a route cannot be determined locally to the gateway.

5.7 Coping with emergencies

Sometimes, even for a mail manager responsible only for what happens on his own host, a critical situation can arise when problems have to be dealt with in real-time. For gateway managers, the situation is even more complicated since they have to solve problems related to multiple E-mail systems using different protocols and on which they have no control at all.

When some types of problems occur, gateway managers have to react quickly to ensure that the whole gateway system does not collapse. For example, when messages start to loop (see section 4.3) or when a message queue is blocked, quick manual intervention is necessary, otherwise the gateway gets saturated and may loose irremediably messages still arriving that cannot be treated because of lack of resources or complete blocking.

5.8 Administrative issues

The operation of an E-mail gateway cannot be considered simply as a localized and independent activity since by definition, a gateway is used to interconnect existing E-mail systems. This implies that some external factors, on which the gateway implementors usually have no influence or means of action, have to be taken into account.

For example, the gateway must comply to the regulations imposed by the PTT's which currently have a monopolistic attitude for all that concerns telecommunications. Especially, a gateway is in principle not allowed to switch third-party traffic. Of course, there are lots of subtleties in the matter, and it is not always very clear what is allowed and what is not.

There needs to be some kind of name management across networks to avoid clashes when disjoint networks are connected (see section 4.4), and a decision has to be made when there are several ways to convert from one address format to another.

A gateway must also comply with the rules applied to the networks it interconnects. For example, a gateway host linked to EARN/BITNET will have to use leased lines to connect to that network, unless its nearest neighbour is on the same site.

Another task of the gateway manager is to take care of the machine supporting the gateway, by considering maintenance contracts and back up procedures.

5.9 Conclusion

To conclude this description of the tasks linked to the operation of an E-mail gateway, let's listen to what one of CERN's gateway managers says about her job : "Operating a mail gateway service involves much more than just installing the software. The overhead is a function of the number of gateways, the number of connections, the size of the user community and the quality of the mail and gateway software - not only run locally but also at the external sites users are communicating with." [Heagerty87a].

Chapter 6
A particular case of E-mail gateway system : CERN

Now that we have learned a little more about the general principles surrounding E-mail gateways, it is time to see how all these ideas are applied in real life. In order to do this, the gateway system set up at CERN, the Organization for Nuclear Research in Europe located in Geneva, will be described.

After having stated the initial needs and the preparatory work concerning the gateway system that was to be set up, we will have a closer look at the E-mail systems that were to be interconnected. Two software products constituting the heart of the gateway will then be described, as well as the machine at the center of the system.

Then, the gateways to the major E-mail systems will be described individually. A special type of gateway will also be mentioned, namely a gateway from E-mail systems to the Telex network. After having said a few words about CERN's directory service, we will close this chapter by describing the trend for the future of E-mail at CERN, i.e. the migration to X.400.

6.1 Initial needs

CERN is one of the biggest research laboratories in the world. As such, it employs about 3500 persons, a third of which are engineers and scientists. In addition to the staff members, about 3000 physicists coming from all over the world work on the CERN site but keep frequent contacts with their home country. All these people are part of the High Energy Physics (HEP) community which is characterized by a large dispersion worldwide and by high communication needs.

In 1985, the situation was the following. CERN was linked to the rest of the HEP community via wide area networks having each their E-mail protocols. There were connections to JANET (the UK Joint Academic NETwork using Grey Book, a modified version of RFC 822, as mail protocol), to INFNET (the Italian HEP private DECNET, using VMS Mail), to EARN (the European counterpart of BITNET increasingly using RFC 822) and to EUNET (the European Unix NETwork using mainly RFC 822).

Moreover, inside CERN itself, there were, in addition to the E-mail systems used on the networks mentioned above, other mail systems, including NOTIS-ID, on Norsk Data machines and EMF, a set of Electronic Mail Facilities developed at CERN and used on an IBM/MVS host. All these E-mail systems were incompatible and many CERN hosts had no E-mail system installed at all.

The situation was clearly unsatisfactory, since, on the one hand data communication facilities had become such an important tool for the daily work of the HEP community, and on the other hand, E-mail as a data communication tool is only useful if every member of a group can reach any member of any other group, which was not the case.

However, it should be noted that partial gateways between the networks to which CERN was connected and also to other networks to which CERN had no direct connection already existed abroad. For example, there was a gateway between BITNET and UUCP in the USA, and it was possible to get indirectly connected to ARPA Internet or CSNET via BITNET.

But this is far from being sufficient when huge amounts of messages have to be regularly treated, as is the case for CERN and the HEP community. The use of gateways indeed implies additional delays and costs and possibly some loss of functionality. Furthermore, the same connectivity and effectiveness as the one achieved with a local gateway system could not be provided, in particular for the local users.

Likewise, it would be ridiculous to have to send a message via the United States to go from a BITNET host at CERN to, for example, a UUCP host in Belgium.

6.2 The COMICS study

In order to solve the E-mail problems at CERN, a study named COMICS (COmputer based Message systems InterConnect Strategy) was carried out during the year 1984.

Two major recommendations of COMICS are summarized as follows [Beyschlag85a] :

- "X.400 compatible Electronic Mail Systems should be installed and used wherever available. All new developments should be done in the context of X.400"

- "A flexible gatewaying system to interconnect X.400, EARN, VMS Mail over DECNET, Grey Book over JANET, EUNET and other Electronic Mail Networks and Systems should be supported on a central mail server. [...] This central system would be replaced gradually by an X.400 network with gateways into other networks. [...]"

(See section 6.11 for a discussion on the choice of X.400 as standard E-mail protocol.)

In June 1985, the implementation of the gateway system described in the COMICS report began in the framework of the MINT (Mail INTerchange) project. The goals of this project were to establish the MINT gateway computer, to provide connectivity between recommended mail systems at CERN and to provide a uniform addressing syntax for mail [Heagerty86].

6.3 The strategy model

During the course of the COMICS study, two gatewaying models emerged [Beyschlag85a]. In the first model (centralized gateway), an integrated E-mail server would contain the gateways between all considered E-mail systems, either on a one-to-one basis or integrated with a central meta-protocol.

This model has the advantage of being easier to control. The gateway can be tuned according to the needs of the user community. For example, calling frequencies and line speeds to neighbouring hosts can be adapted to the mail volume quite easily. Statistics on the use of E-mail are easier to obtain because there is only one point where messages cross the border between E-mail systems [Beyschlag87a]. When several gateways are involved, as in the current situation, it is not always easy to ensure that messages are counted only once in the statistics.

The disadvantage of such an approach is that all messages that have to be gatewayed must pass through a single computer. This may cause reliability and capacity problems.

The second model (distributed gateways) is based on multiple gateways that are used wherever they exist. In such a model, the gateways can be maintained by the experts of the mapped protocols themselves.

The strategy finally chosen is a mixture of the two models. It recommends a centralized gateway for the short term future. X.400 would be one of the gatewayed networks. This configuration would soon develop into a distributed gateway approach where X.400 as the protocol with the highest functionality is placed in the middle and gateways at CERN and outside CERN are used to connect to E-mail systems with other protocols (see fig 6.1).

This approach has the advantage of giving a special importance to the X.400 system, since the only remaining gateways will eventually be between X.400 and other E-mail systems. This will give X.400 a central role and ease the transition to X.400, as is planned for the medium to long term future.

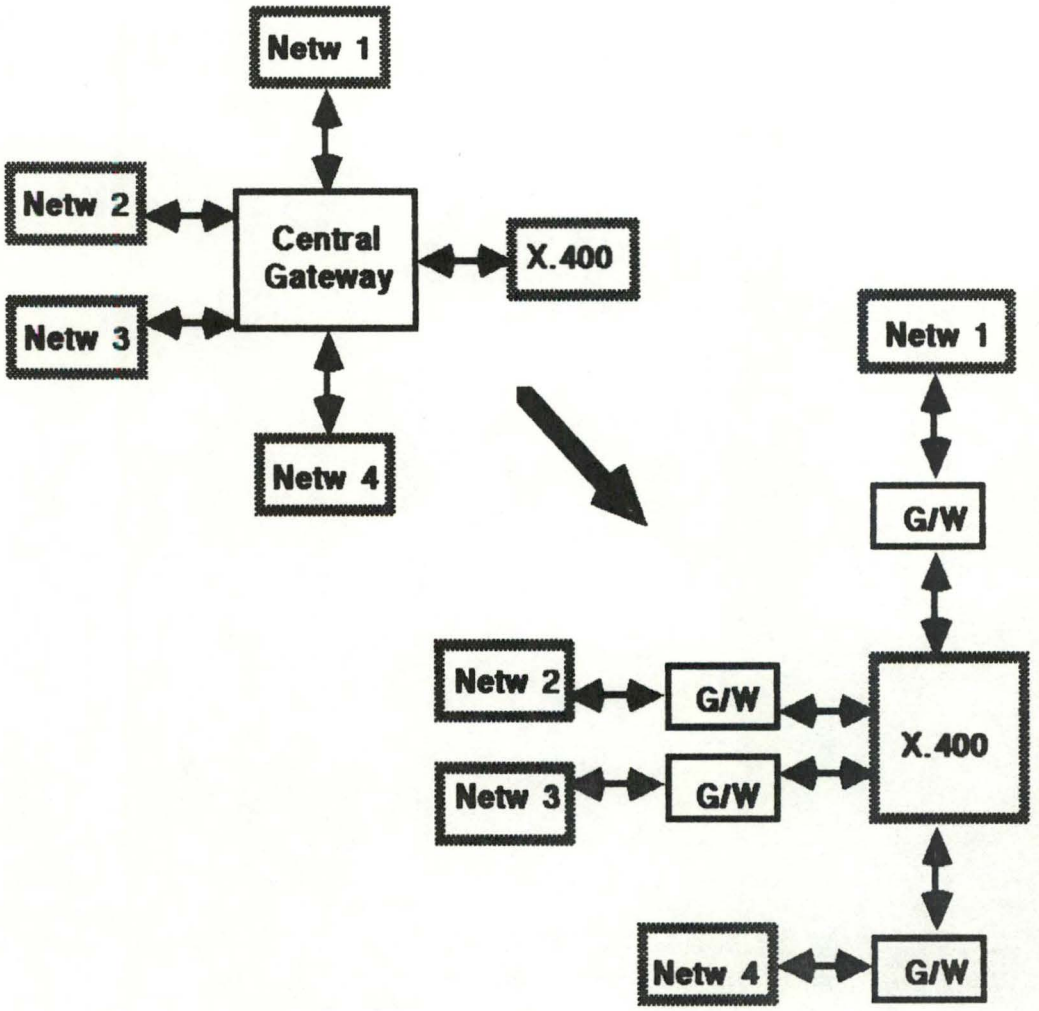


Fig 6.1 : Gateway model evolution

6.4 Major E-mail systems used at CERN

At the end of the COMICS study, lots of different E-mail systems were in use at CERN. Sometimes, several of them were used on the same machine. So, to ease a little the life of everyone, it was decided to select for each type of machine and operating system an E-mail system which would become the recommended one (see table 6.1).

Using a recommended E-mail system implies the following advantages. They are supported on the CERN site, i.e. there is at least one local specialist able to answer questions associated with a recommended E-mail system, and to solve related problems. The probability is also high that there is a gateway between any two recommended E-mail systems, while it is not so sure for others.

Moreover, the fact that EAN was chosen as the recommended E-mail system against VMS Mail on VAX/VMS and against the Unix mail on VAX/Unix will clearly facilitate the transition to X.400, which is actually the long term goal. EAN is currently running on 10 VAX/VMS and 2 VAX/Unix machines at CERN.

Machine/Operating System	Recommended E-mail system
VAX/VMS	EAN
VAX/Unix	EAN
IBM/MVS	Wylbur EMF
IBM/VM	MIT/Rice Mail Exec & Columbia mailer
Norsk Data/Sintran	Notis-ID

Table 6.1 : Recommended E-mail systems at CERN

Even though they are not mentioned in table 6.1, it is also possible to use Unix mail, which is the standard mail program available on Unix systems and VMS Mail, which is also the standard mail facility shipped with VAX/VMS machines. There are currently about 200 CERN hosts and 1500 hosts around the world that use VMS Mail as first communication means.

It should also be noted that the central IBM/MVS service is planned to be stopped soon altogether and that this service is progressively replaced by another IBM operating system, namely VM/CMS.

The Norsk data machines and their E-mail system Notis-ID are used mainly for CERN's internal administrative needs, and little, if at all, by the physicists for their external contacts.

These reasons explain why gateways to and from the IBM/MVS and the Norsk Data machines will just be mentioned, while gateways to the UUCP world and to VMS Mail systems will be described, even though these E-mail systems are not recommended ones.

6.5 Two products of particular importance

The heart of the gateway system set up at CERN is constituted by two products. On the one hand, there is sendmail, which is a general routing program to which several E-mail systems can be interfaced. On the other hand, EAN is the X.400 implementation on which CERN has based its long term migration strategy.

6.5.1 Sendmail

Sendmail implements a general internetwork mail routing facility, featuring aliasing and forwarding, automatic routing to network gateways, and flexible configuration [Allman83]. It is part of Berkeley Unix and has been designed with RFC 822 in mind. One of its main goals was to ease the transition from the traditional UUCP "bang path" addressing to domain addressing in the ARPA Internet style.

Here follow some of the characteristics of sendmail. Each processed message is guaranteed to be properly delivered or forwarded, and at least not lost. Sendmail is driven by a configuration file read at each invocation, which allows to change most parameters (e.g. parsing rules or routing information) without recompilation. Network traffic is minimized by automatically batching addresses to a single host where possible. It is also possible to send mail directly into a file or as input to a command.

a. Communications with the outside world

Sendmail has generally no direct contact with users and does not perform actual mail delivery. Rather, it collects a message generated by a user interface program which will act as the sender, edits the message as required by the destination E-mail system, and calls the appropriate mailer(s) to do mail delivery or queueing for network transmission (see fig 6.2).

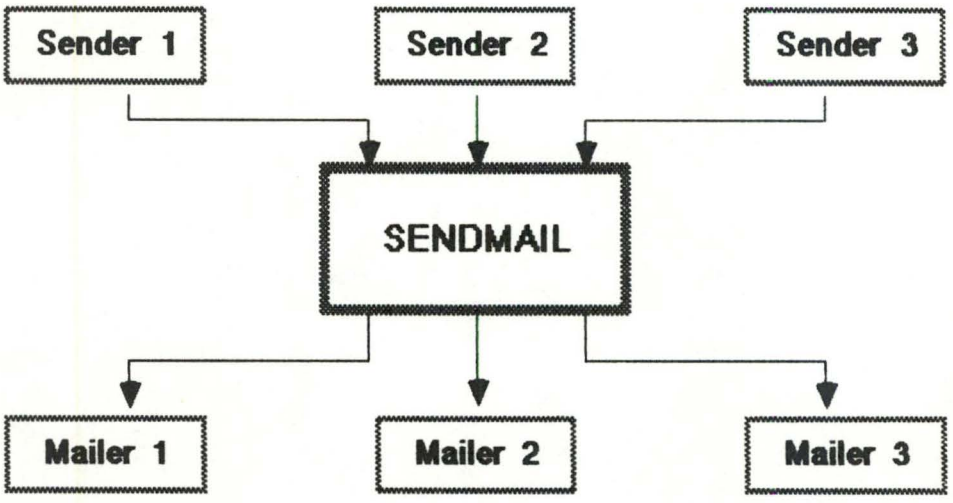


Fig 6.2 : Sendmail's interactions with senders and mailers

There are three ways sendmail can communicate with the outside world, i.e. with the sender and mailer programs it interacts with. These three techniques can be used by the senders to transmit messages to sendmail, as well as by sendmail to pass messages to the mailers.

The first one is by using the standard means for communicating with Unix processes, i.e. by using the argument vector/exit status technique. Here, the recipients of a message are passed as parameters in the argument list and the message body is presented at the standard input. The exit status of the receiving process is collected and appropriate action taken if necessary (e.g. an error message is sent to the sender of an undeliverable message).

The second approach is to speak SMTP over a pair of Unix pipes. Recipients are not passed in the argument list. Instead, a usual SMTP dialog takes place, using the standard input of the receiving process to pass SMTP commands, and collecting the reply codes on the standard output of the same process.

The third possibility is to talk SMTP over an InterProcess Channel (IPC). This method is quite similar to the second, except that it is much more flexible, due to the use of 4.2BSD IPC, which is generally used to allow communication between processes on different machines.

b. Typical scenario

When a sender wants to send a message, it issues a request to sendmail using one of the three methods described above. Sendmail processes this message in three phases. First, it interprets the arguments and parses the addresses, then it collects and stores the message, and finally the message is handed to the appropriate mailer for delivery.

During the parsing of addresses (coming either from the argument list or the SMTP command RCPT), a list of recipients is created. Address interpretation is controlled by a production system, which can parse both RFC 822 domain-based addressing and old-style bang path addresses and even most mixtures of these. As much verification as possible of address syntax is done at this step.

Aliasing and forwarding are also done here. Aliasing is the replacement of alias names with the corresponding list of addresses, using a system-wide file. Forwarding allows each recipient to specify (in a file in his home directory) a list of users to which any message should automatically be forwarded.

Then, the message is collected from the sender. The header is parsed and kept in memory while the message body is stored in a temporary file.

The recipient list is then rearranged according to the mailers that are to be invoked. For each recipient, sendmail knows which mailer it has to call thanks to the format of the address or the names of host or domain contained in the address.

Each mailer is then passed the list of recipients it will have to take over, using one of the techniques described above. Then the message itself is sent. Sendmail makes the per-mailer changes to the header (e.g. to ensure that it will be possible to use automatic reply), if necessary, but the message body is passed untouched.

If a mailer returns a status indicating that delivery is currently not possible, sendmail will queue the message for retransmission and retry later.

c. The configuration file

At each invocation, sendmail reads a configuration file containing the information it needs to parse addresses correctly and to find the mailers to which it has to pass the messages. The configuration file is composed of header definitions, mailer declarations, address rewriting rules, macro definitions and options.

Header declarations specify the format of header lines to be added by sendmail if necessary. Mailer declarations indicate to sendmail what mailers are available to it and when to use them, with what parameters, etc. The heart of the address parsing of sendmail is constituted by a set of rewriting rules comparable to a production system. This allows a very flexible (but not always easy to code and to read) editing of addresses.

6.5.2 EAN

In view of the long term goal which is the migration towards X.400 conforming systems and the eventual eviction of all other E-mail systems, the adoption of EAN can be considered as a key step. EAN (version 2) has indeed become the recommended E-mail system on Unix as well as VMS VAXes at CERN.

The integration of this X.400 E-mail system in CERN's gateway system will allow everyone to migrate towards X.400 at his own pace, while at the same time ensuring the same connectivity for all sites, even the ones that can adopt X.400 only at a slower pace.

The EAN package includes interfaces to DECNET, to TCP/IP and to the DEC PSI X.25 software, as transport mechanisms. The X.400 messages of EAN are thus transferred using the P1 protocol over DECNET between VAX/VMS systems, over TCP/IP between VAX/VMS and VAX/Unix and over the Public PSDN between other EAN/X.400 PRivate Management Domains.

6.6 A special machine : CERNVAX

Following the COMICS study, the MINT (Mail INTerchange) project was started with the aim of providing a central E-mail gateway computer. As will be seen in the next section

(6.7 Gateways implementation), the gateway system is implemented using several machines. However, two of these, MINT and PRIAM, have a special status. Maybe their first characteristic is that they both have the same second name, CERNVAX. Here is their story.

It all started out with a VAX11/780, with a UUCP connection to MCVAX, the Dutch (and European) UUCP backbone host. The UUCP host name was CERNVAX. This machine provided at CERN the service for a project called "PRIAM" (PRoject Interdivisionnel d'Assistance aux Microprocesseurs). Then, the COMICS study started and with it came the first version of X.400 based mail, i.e. EAN. The EAN host, the same VAX, was called priam. Soon after that, CERN was connected to EARN/BITNET, and the UREP software (Unix Rscs Emulation Package) was installed on that machine too, with CERNVAX as EARN hostname.

Then the decision was taken to install a machine dedicated to E-mail gatewaying, MINT, and the EARN/BITNET connection was moved to that machine, a VAX11/750, to relieve the overloaded VAX11/780 (PRIAM) of some of its burden. The 780 has been replaced by a much more powerful VAX 8530 last year, but the mail connections have not changed, so CERNVAX (MINT) is on EARN/BITNET and CERNVAX (PRIAM) is on EUNET/UUCP.

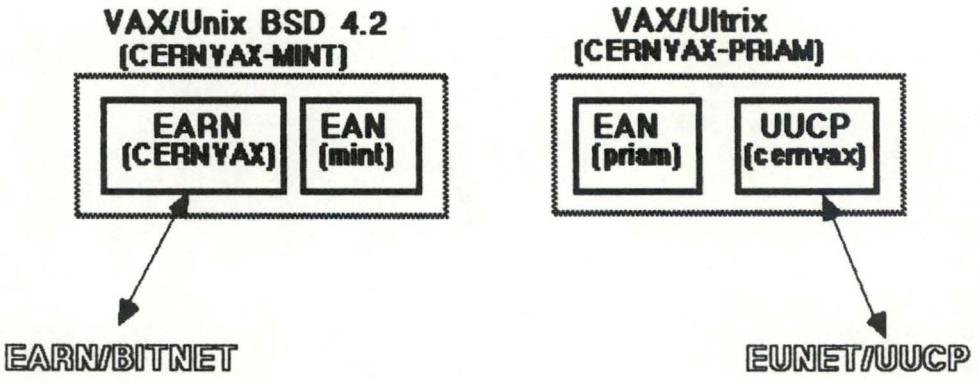


Fig 6.3 : The CERNVAX machine(s)

Figure 6.3 shows both CERNVAXes, the names of the E-mail systems of which they are hosts, and the names under which they are known in the different E-mail systems. The interactions of the functional components shown in that figure as well as how they fit in the gateway system are explained in detail in section 6.7. The two arrows indicate that there is a direct connection to, respectively, the EARN/BITNET and EUNET/UUCP networks, i.e. that both CERNVAXes are the entry/exit points between CERN and those

networks. Even though the EAN software is present in both of them, the corresponding entry/exit point to EAN is located in another machine, explaining the absence of an arrow to the EAN network.

To lift the ambiguity about messages addressed from EARN/BITNET to a user on CERNVAX, some trick had to be used, since two machines respond to that name. BITNET transfers are file transfers, and every transfer consists of a data part and a tag part. The tag contains the information on remote and local user and host, i.e. sender and recipient. In E-mail, terms, we would call the tag a message envelope. The mechanism implemented in MINT checks for the destination user being a genuine PRIAM user (the list is updated once per day), and in that case does not interpret the data part of the BITNET file, but sends it as mail to the specified user. Otherwise, it takes origin and destination from what it finds in the data part. This can be a message body with an RFC 822 header with or without an SMTP envelope. The sendmail configuration file on MINT contains the information on what to do with mail to be sent to non-CERN hosts on other networks. EAN-bound mail and DECNET mail are sent to a VAX/VMS machine (VXGIFT) and UUCP mail is sent to PRIAM, using TCP/IP over Ethernet in all cases.

6.7 Gateways implementation

In this section, the gateways operated at CERN to and from the four major networks will be described. Possible alternatives will also be mentioned.

6.7.1 EARN/BITNET gateway

To set up a gateway to EARN/BITNET, all that is needed is an interface from this network to sendmail. Indeed, when this interface is achieved, a message coming from EARN/BITNET can be passed to sendmail and rerouted by this to EAN or EUNET/UUCP, which are also interfaced to sendmail.

A message can then even be forwarded to any other network which can be reached via EAN or EUNET/UUCP. For example, if the message is to be sent in the DECNET world, it will first be passed to EAN which will forward it to its final destination (see section 6.7.4 for details on that gateway).

Likewise, any message that was somehow passed to sendmail, either coming from EAN, EUNET/UUCP or indirectly from any E-mail system connected to these, can be redirected to EARN/BITNET.

The passage between EARN/BITNET and sendmail, which is in fact the critical point of the gateway, will not imply any loss of functionality once RFC 822 will be used as the only end-to-end protocol on EARN/BITNET.

The implementation of this gateway involves two hosts at CERN (see fig 6.4) : CEARN, which is an IBM/VM system acting as the Swiss International EARN node, and MINT, which was described above. On the MINT machine runs a software called UREP (Unix Rscs Emulation Package), developed by the Pennsylvania State University. Its goal is, as its name indicates, to emulate on a Unix machine the RSCS (Remote Spooling Communications Subsystem) protocol used on IBM systems. An interface to sendmail is included in the package. The BSMTP protocol (see section 2.2.2) is used as mail transfer protocol over RSCS.

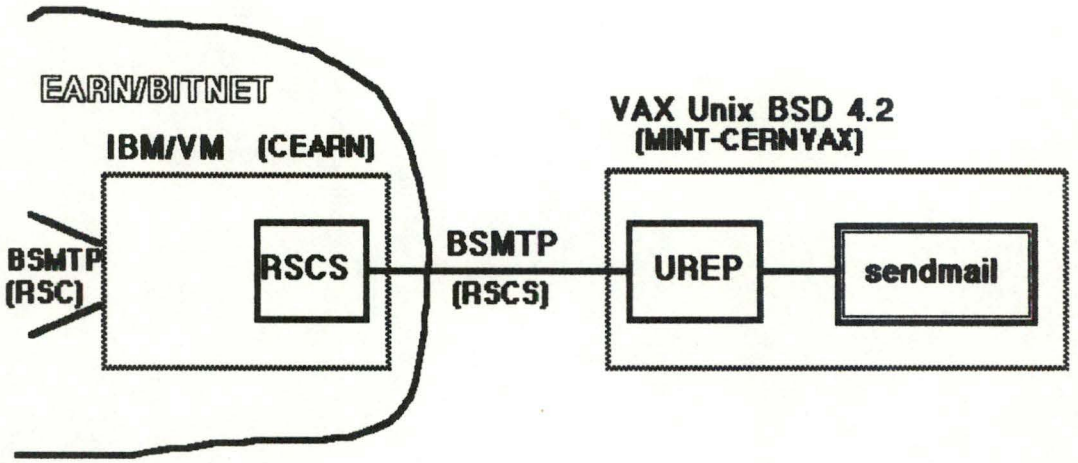


Fig 6.4 : The EARN/BITNET gateway

Independently of any E-mail consideration, UREP makes it possible for a Unix machine to become a peer host fully integrated to the EARN/BITNET network. This permits a Unix user to submit jobs for remote batch execution or transmit files to or from any EARN/BITNET node.

A detailled description of the journey of a message passing from EARN/BITNET to EUNET/UUCP is given in section 6.8.

An alternative to the UREP solution would have been to use the Wiscnet software developed by the University of Wisconsin. This package allows an IBM/VM to communicate with a Unix machine, using TCP/IP over Ethernet or X.25, and using SMTP as mail transfer protocol.

6.7.2 EAN gateway

The integration of EAN to existing E-mail systems is done at two points. The first one consists of an interface to sendmail, included in the Unix version of EAN. This allows any EAN message to be passed to sendmail, and from then on, to any E-mail system interfaced to sendmail, including EARN/BITNET and EUNET/UUCP. This of course works also in the other direction.

This interface to sendmail consists of some code, included in the EAN package itself, to convert the X.400 data structures used by EAN (with the restrictions mentioned in section 2.4.4 since EAN is not a pure X.400 system) into the RFC 822 format that sendmail can understand. The sender part is called instead of the usual EAN transport program when the address of a message contains a domain which has been declared as special domain in EAN MTA tables.

Likewise, there is a "mailer" part of the package to convert from RFC 822 to X.400 format. This mailer is called from sendmail when the latter has decided, following the rules contained in its configuration file, to send a given message to the EAN network.

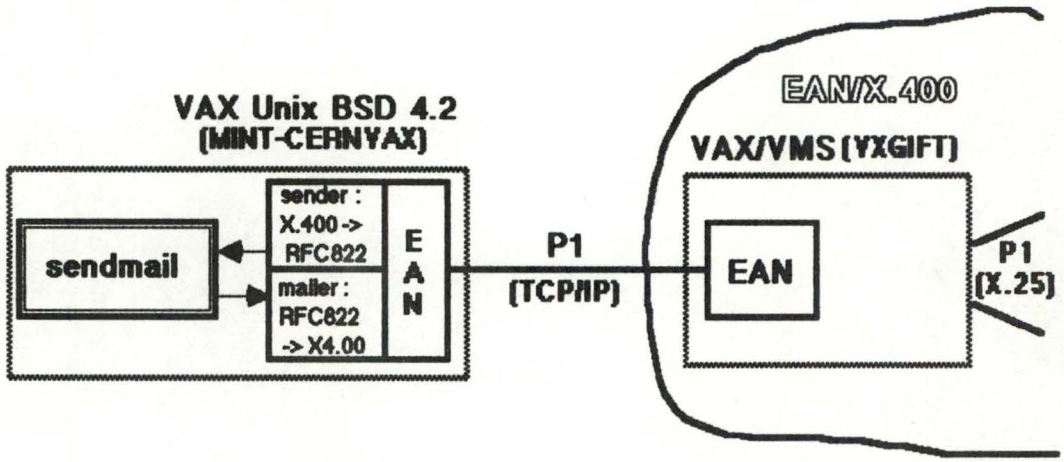


Fig 6.5 : The EAN gateway via sendmail

This passage from P2 to RFC 822 implies of course some loss of functionality. For example, probes cannot be treated properly, and confirmation of delivery are not part of "standard" RFC 822.

At CERN, the EAN gateway is once again hosted in the MINT-CERNVAX machine (see fig 6.5). All the external EAN traffic is passed to a VAX/VMS (VXGIFT), which is the entry/exit point to other EAN/X.400 domains, via X.25 lines.

A gateway between EAN and VMS Mail is also possible, as will be seen in section 6.7.4 related to VMS Mail.

6.7.3 EUNET/UUCP gateway

The key component of the gateway to and from EUNET/UUCP is, once again, sendmail. The Unix mail package is already interfaced with sendmail to ensure proper delivery of mail throughout EUNET/UUCP, independently of any other E-mail system. So, it is quite easy to set up a gateway to other E-mail systems when these systems offer an interface to sendmail, like EAN and EARN/BITNET.

At CERN, the machine which constitutes the entry/exit point to EUNET/UUCP is CERNVAX-PRIAM (see fig 6.6). The sendmail program hosted there is in direct connection with its counterpart sendmail on the central gateway machine, CERNVAX-MINT.

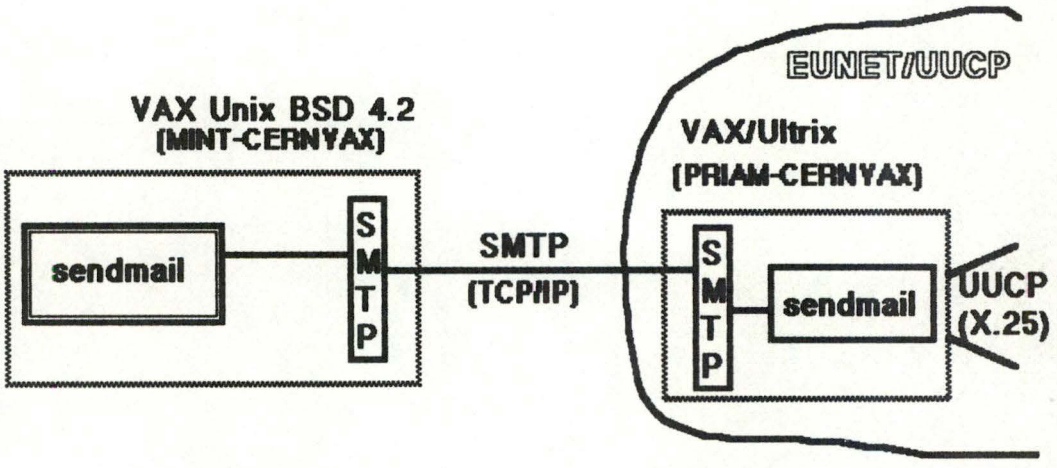


Fig 6.6 : The EUNET/UUCP gateway

Once a message coming from EUNET/UUCP has been passed to sendmail on CERNVAX-PRIAM, it is passed, if it is not for this host, to sendmail on CERNVAX-MINT. This happens using SMTP over a TCP/IP link. Once there, the message can be forwarded on any E-mail system already interfaced to sendmail, just by having made sure that sendmail will pass the message to the proper mailer. This works likewise in the other direction.

6.7.4 DECNET gateway

The VMS version of EAN includes an interface to VMS Mail. This gateway code allows to convert the X.400 data structures used by EAN to and from a form compatible with the VMS Mail and vice versa.

In the VMS to EAN direction, the gateway makes use of the foreign mail protocol feature of VMS Mail. This characteristic allows an address to have the form

```
pgm%"address-part"
```

When the VMS Mail program has to treat such an address, it calls (in fact, it links dynamically) the program named before the "%" sign, and passes the control to this, with "address-part" as argument.

Thus, in this case, an address like

```
EAN%"user@host.domain"
```

would cause a program having a name in the style "VMS_TO_EAN" to be searched on the disk and the control passed to it, with arguments including the actual EAN address ("user@host.domain") and the message, headers and body. The "VMS_TO_EAN" program can then try to map the VMS Mail headers as well as it can and pass the result to the main EAN program which will do the rest of the job.

At CERN, the actual format used is 'MINT%"user@host.domain"', MINT being the central gateway machine. The machine where the gateway between VMS Mail and EAN is running is VXGIFT. So, any machine being on the HEP DECNET, like VXGIFT, can reach any host accessible from EAN at CERN, by first specifying the path to reach VXGIFT via DECNET, and then by specifying the address by which the destination host would be known from EAN. Note that the destination host can be on any reachable network, e.g. BITNET or EUNET.

In the other direction, i.e. from EAN to VMS mail, things happen differently. To activate the gateway possibility of EAN, an MTA named "vmsmail" has to be declared in the MTA tables. A domain name is associated to this MTA, so that each time a message is queued for a user on a host in that domain, a special gateway program is invoked instead of the usual program to transfer messages between MTA's using P1.

In the case of CERN, the name associated to that special MTA is DECNET.CERN, so that each time an EAN message as the form

user@decnethost.DECNET.CERN

the message is gatewayed to VMS Mail.

Figure 6.7 shows CERN's implementation of the gateway, in both directions.

Loss of functionality is guaranteed in most cases since the protocol used by VMS Mail is a much more primitive end-to-end protocol than X.400. There is no probe, no confirmation and even no carbon copy, just to mention these.

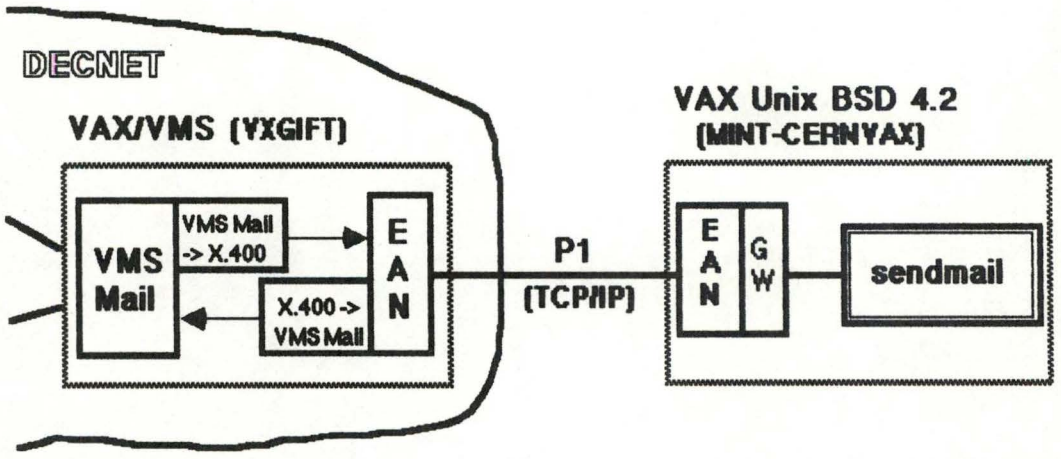


Fig 6.7 : The DECNET gateway

The EAN/VMS Mail gateway described hereabove is not satisfactory for two main reasons [Heagerty87d]. First, there is no retry mechanism, so that when a message comes from EAN to DECNET and a direct connection to the DECNET host is temporarily unavailable, the message is simply discarded and a non-delivery notification is returned to the sender. The second reason is that error messages do not indicate clearly the reason for message failure to a DECNET node.

For these reasons, the current gateway will soon be replaced by a new one. The most likely alternative gateway will be implemented using the PMDF (Pascal Memo Distribution Facility), which is a routing facility comparable to sendmail and running on VAX/VMS. It has interfaces to VMS Mail, to JNET (see below) and provides store and forward capability for DECnet. It could also be interfaced to sendmail on the central gateway machine (see fig 6.8).

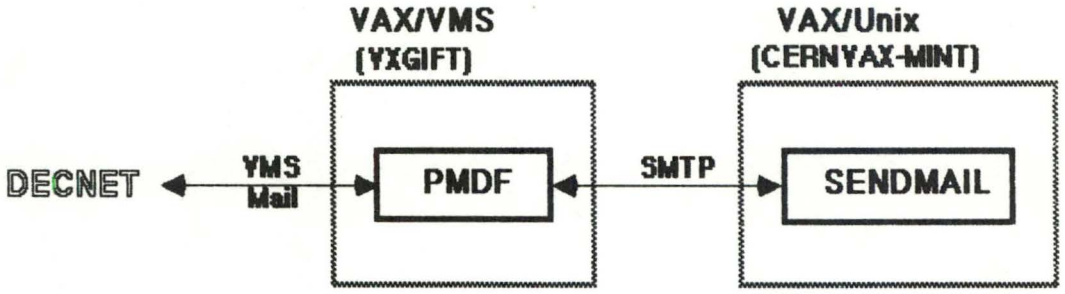


Fig 6.8 : The alternative DECNET gateway

Yet other solutions exist, at least for what concerns a direct gateway between DECNET and EARN/BITNET. There exist indeed two products, JNET and INTERLINK, used as emulators of EARN/BITNET protocols. JNET allows a VAX/VMS machine to appear as an EARN/BITNET host, and emulates the RSCS protocol, as does UREP on Unix machines. INTERLINK permits an IBM/VM host to be part of DECNET.

DECnet runs also under the Ultrix operating system, DEC's Unix product, providing yet another, albeit somewhat deficient gateway facility.

6.7.5 Global view of the gateway system

Figure 6.9 shows a global view of the gateway system involving the four major networks. On this figure, it can be seen that the gateway system used at CERN is supported by four machines, having each a particular role.

CEARN, an IBM/VM machine which is also the Swiss international EARN node, is the entry/exit point to and from EARN/BITNET. VXGIFT, a VAX/VMS, is simultaneously the entry/exit point to and from the international EAN/X.400 networks and the HEP DECNET. CERNVAX-PRIAM, a VAX/Ultrix, is the entry/exit host to the EUNET/UUCP world. CERNVAX-MINT, a VAX/Unix, is the central gateway machine responsible for the exchange of messages between the networks mentioned above.

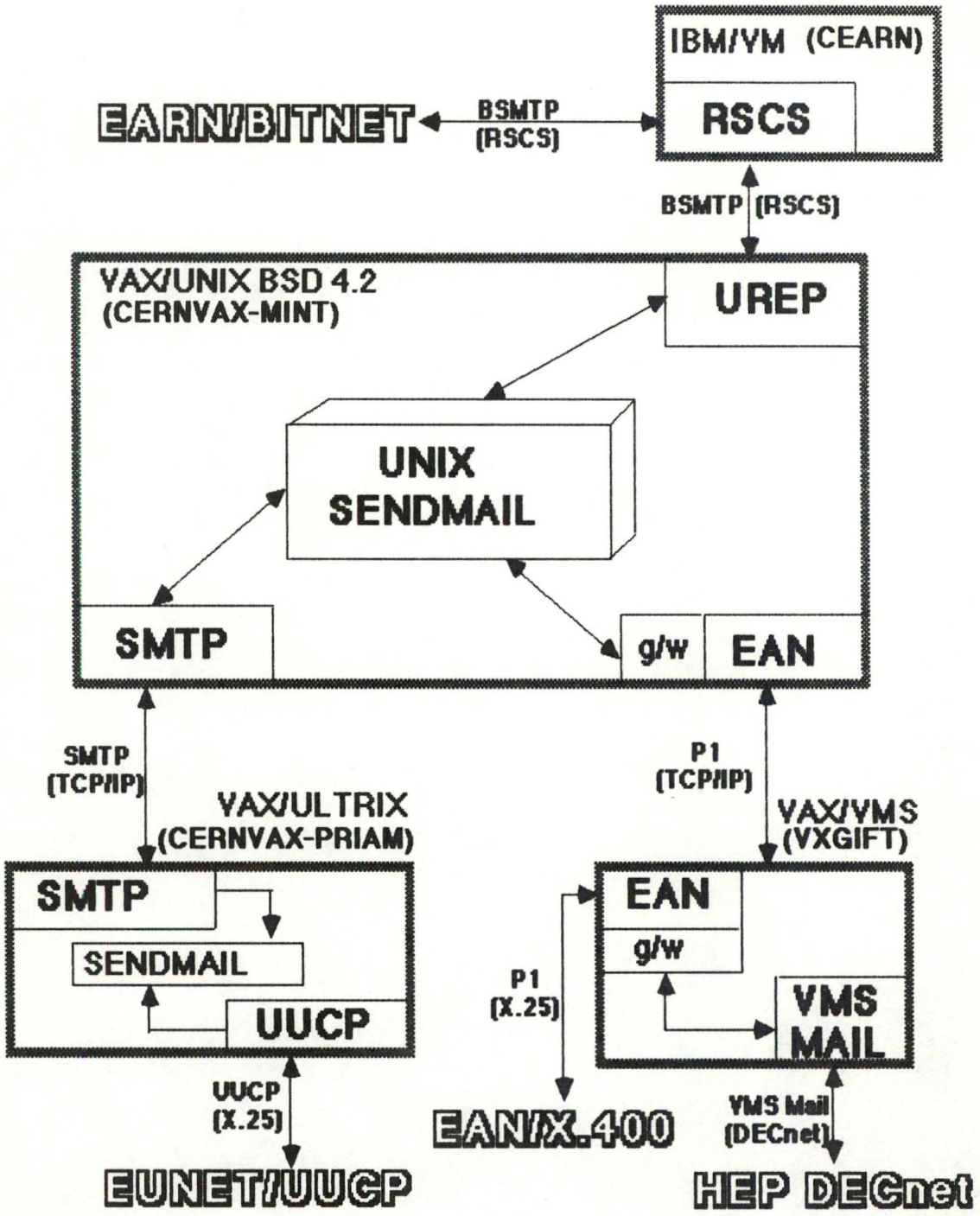


Fig 6.9 : Global view of the gateway system

6.7.6 Other gateways

Apart from the major networks, namely EARN/BITNET, EUNET/UUCP, EAN/X.400 and DECNET, other E-mail systems are involved in the gateway system at CERN.

EMF (Electronic Mail Facility), is a collection of functions written at CERN to allow mail to be exchanged between the IBM/MVS machine and EARN/BITNET (and from there to other E-mail systems as well). The IBM/MVS is being replaced by an IBM/VM (CERNVM), but once was the central IBM machine and had even been considered as a possible host to implement a central mail facility where every user at CERN and in the HEP community would have had its own mailbox. That solution was rejected in favor of the current more flexible and effective gateway system.

On an other hand, for the moment there is at CERN no gateway to JANET, the UK network. Messages are currently exchanged between JANET and CERN using a gateway between EAN and Grey Book at ULCC (University of London Computer Center) and a gateway between EARN/BITNET and Grey Book at RAL (Rutherford Appleton Laboratory). However, a local gateway to JANET is planned.

As for the Notis-ID E-mail system, a rudimentary gateway to the central gateway facility is planned, but as soon as X.400 products for Norsk Data machines will be available, they will be used to interconnect to the other E-mail systems.

6.8 Example of the journey of a message

To have a better idea of the precise steps a message crossing a gateway has to pass through, let's follow a message sent from userX on earnhost, a hypothetical EARN/BITNET node, to userY on fun-cs, one of the EUNET/UUCP hosts on our site.

```
9   From prlb2!cernvax!EARNHOST.BITNET!userX  Tue Feb 10 09:50:48 1987
8   Received: by prlb2.UUCP (4.12/4.7)
      id AA09517; Mon, 9 Feb 87 14:35:32 +0100
7   Received: by cernvax.UUCP (4.12/4.7)
      id AA01730; Mon, 9 Feb 87 11:59:39 +0100
6   Received: from mint.cern (mint) by cernvax.UUCP (4.12/4.7)
      id AA01727; Mon, 9 Feb 87 11:59:28 +0100
5   Received: by mint.cern (cernvax) (E.12/3.14)
      id AA21295; Mon, 9 Feb 87 11:59:09 +0100
4   Message-Id: <8702091059.AA21295@mint.cern>
3   Received: by cernvax Mon Feb 9 11:59:04
      from userX@EARNHOST.BITNET via rscs.
2   X-Bitnet-Sender: userX@EARNHOST.BITNET
1   Date: Mon, 09 Feb 87 11:56:44 SET
    To: prlb2!fun-cs.UUCP!userY
    Subject: test

    This is the text of the message...
```

Fig 6.10 : Example of EARN to EUNET message

The message as received by userY in his mailbox is shown in fig 6.10 (except for the numbers that where added in front of some lines to ease the following discussion). Let us have a closer look at it. The structure of the message contains a lot of information concerning the journey of the message.

In order to send his message, userX invokes his mail program with the command

```
mail userY%fun-cs.uucp@cernvax
```

using source routing as seen in section 4.2.2, and specifying the EARN/BITNET host cernvax as the gateway machine.

The mail software on earnhost collects the message the user types in and creates RFC 822 header lines indicating sender and receiver of the message, date and time of expedition, and, optionally, a subject. These lines are shown as lines 1 and below in the example message. Message header and message body will be separated by a blank line according to RFC 822. All this will then be wrapped in a BSMTP envelope and sent to MAILER@CERNVAX, using RSCS, the mechanism used to transfer any ordinary file between two EARN/BITNET hosts.

After having possibly passed through intermediate hosts, the message will arrive, as a BSMTP file, on the CEARN machine, connecting CERN to EARN/BITNET. It will then be forwarded to CERNVAX-MINT which is considered, thanks to the UREP package, as an EARN/BITNET host too. It should be noted here that there is no trace of all the intermediate EARN/BITNET hosts that the message passed through.

The mail part of the UREP package on CERNVAX-MINT, upon reception of the file, prepends line 2 to the message, indicating that the message actually comes from EARN/BITNET. This is the original form of the sender address in the message header or message envelope (not in the tag of the file). Line 3 is then prepended too, this time specifying the "from" as indicated in the tag (in this case, it happens to be identical to what is in the header or envelope). UREP then converts the information about sender and receiver in the BSMTP envelope into arguments used in a call to sendmail on the same machine. At that moment, the message quits the EARN/BITNET world to enter the heart of the gateway system.

Sendmail on CERNVAX-MINT receives the message (passed to it by UREP via one the three mechanisms described in section 6.5.1.a) and prepends lines 4 and 5. Using its

configuration file and the local part of the original address of the recipient (i.e. userY%fun-cs.uucp), sendmail detects that the destination user is on EUNET/UUCP and finds the description of the mailer used to send messages to EUNET/UUCP. Hence it forwards the message using SMTP to sendmail on CERNVAX-PRIAM, since CERNVAX-PRIAM is responsible for all external traffic to and from EUNET/UUCP. The message has then entered the UUCP world and will be transferred as any UUCP message. But let us keep following the message.

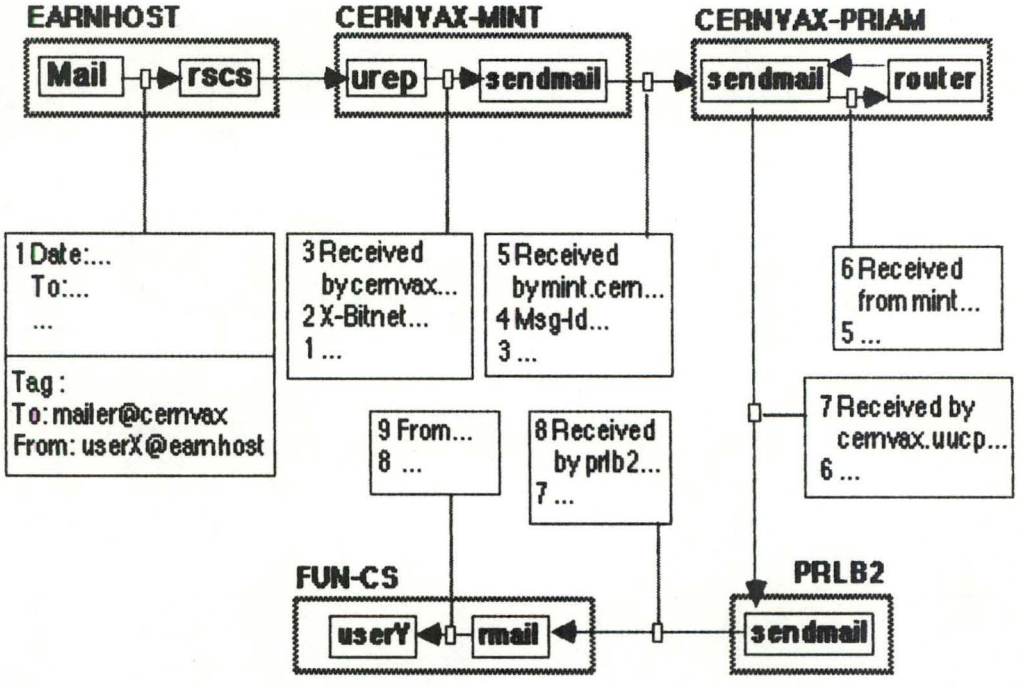


Fig 6.11 : Journey of EARN to EUNET message

Thus, sendmail on CERNVAX-PRIAM receives the message and prepends line 6 indicating the passage of the message from CERNVAX-MINT to CERNVAX-PRIAM. It calls a program implementing a routing algorithm used to determine the national EUNET backbone host to which the message is to be forwarded next. This search is necessary because the destination host, fun-cs, is not directly connected to CERNVAX-PRIAM. The router program then calls sendmail back with a rewritten address indicating the complete route to the destination host (in this case prlb2!fun-cs!userY). This results in line 7 being prepended to the message.

The latter is then queued for transmission to the first host indicated in the new address, i.e. prlb2, which is the Belgian EUNET backbone node. When sendmail on prlb2 receives the message (via the mechanism described in section 2.2.3), it prepends line

8, as would any intermediate EUNET/UUCP host, to provide trace information. Prlb2 in turn forwards the message received from CERNVAX-PRIAM to the next host indicated in the path, in this case the final destination, fun-cs. At fun-cs, the rmail program is run, following the remote execution request of sendmail at prlb2, and since the user is local, the from lines prepended to the message are "folded" into a single line (line 9), and the message is delivered to userY's mailbox.

Figure 6.11 summarizes the journey of the message, pointing out where the different lines containing the trace information were appended to the original message.

It should be noted that the original address of the recipient, which was coded by the mail interface program of the originator user in a "To:" RFC 822 header as

To: userY%fun-cs.uucp@cernvax

was "edited" by the intermediary "mail reformatters" UREP and sendmail to give a final "To" line of the form

To: prlb2!fun-cs.UUCP!userY

6.9 A special gateway : E-mail to Telex

During the end of 1987, the author participated in the installation of a Telex service at CERN on the central IBM/VM machine (CERNVM). This service now allows the VM (authorized) users to send telexes on their own, directly from their terminal, without having to pass through the Telex Office.

The intention is now to have this facility available for all CERN users, not only those having access to VM/CMS. In order to do this, the author was asked to have a closer look at the feasibility of a gateway between the general E-mail service at CERN and the Telex machine (a VM/CMS service machine) responsible for the automatic sending of telexes. Each user having access to an E-mail system could then send telexes as well.

(N.B. : a VM/CMS service machine is a kind of process which executes in the background and which is normally used to act as a server or to provide a batch service).

The achievement of such a gateway could not be completed, mainly because of lack of time, but the preparatory work lead to the following conclusions.

The simplest solution appeared to be to create on CERNVM a service machine which would act as the gateway (see fig 6.12). The job of this service machine would be to run disconnected under VM/CMS and sleep until an E-mail message addressed to it arrives in its READER (kind of mailbox on VM/CMS). It would then consider the message body as the text of a telex, possibly after some checks to figure out if that was really meant to be a telex.

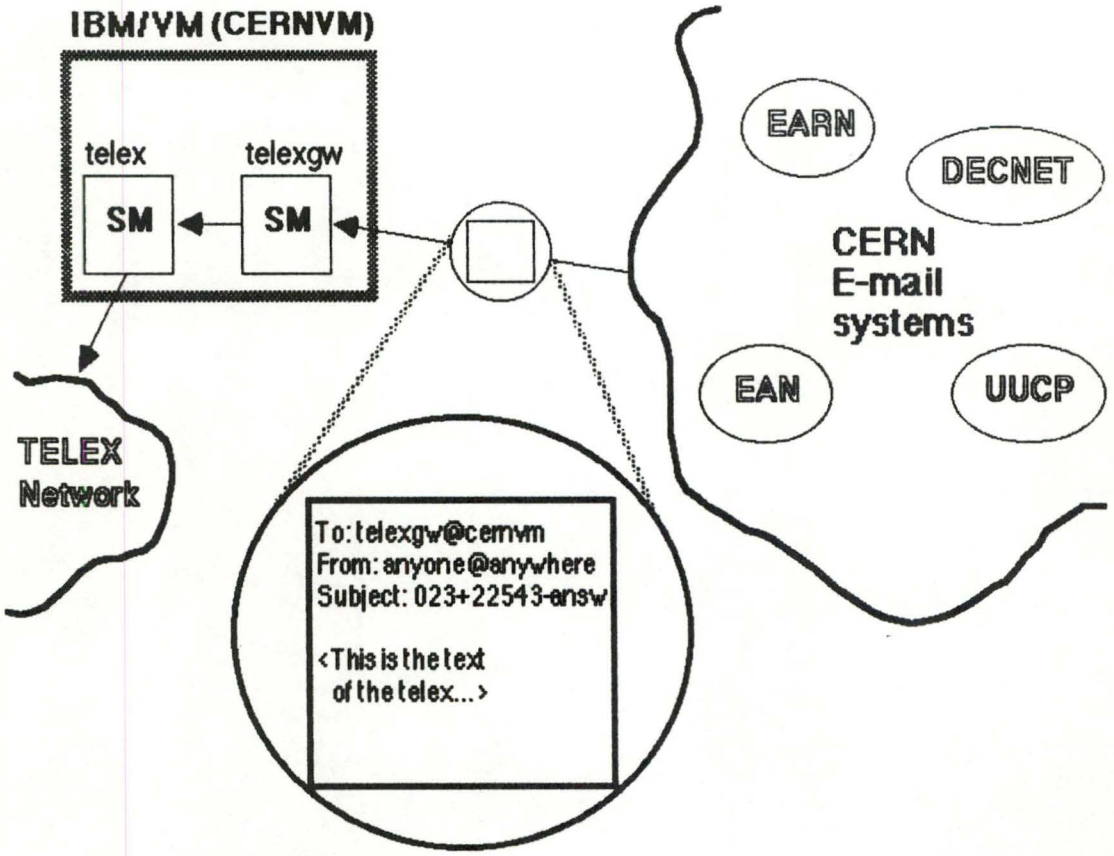


Fig 6.12 : The E-mail to Telex gateway

From the origin, subject and body of the E-mail message, a Telex would be built following the format expected by the service machine sending telexes on the Telex network.

To send a telex, one would thus send an E-mail message to this kind of Telex server, specifying the destination of the telex in the subject part of the E-mail message. A Telex address must conform to the syntax CCC-LLLLL-ANSW, where CCC is the numerical code of the destination country, LLLLL is the Telex number and ANSW is the answerback, which is used to check that the machine at the other end of the line really is the machine that is supposed to be there.

While being attractive because of the apparent ease of implementation, the simplicity of this method, i.e. nothing in the body of the message other than the text of the Telex to send, could hide some possible problems. One of these would be the impossibility for the sender to use options other than the default ones, for example to specify the telex header to use, the urgency of the Telex, what to do with telexes containing syntactical errors, etc.

Another problem would be the difficulty to specify several destinations, in particular when these are numerous, since the subject field in most E-mail systems is of limited length.

Some other problems arise, independent of the method used to specify addresses and options. The authorization problem remains complete, mainly because of the legal importance of telexes. But it appears that the authorization procedure used currently on the Telex machine could be customized and reused without much effort.

Yet another problem is the one of the strict format of telexes. This, in fact, is a special case of body conversion problem, as seen in section 3.4. When sending a telex from VM/CMS, the user has an interactive interface at his disposal, pointing out where the telex could be syntactically incorrect. The characters of a telex come indeed from a character set much more limited than the one available on computer terminals. Moreover, each line of a telex must be smaller than 60 characters. However, when sending a telex from an E-mail system, this kind of protection is not available.

Some of the possibilities to cope with this problem are :

- to simply reject at the gateway every incorrectly formatted telex, i.e. to send an error message back to the sender of the E-mail message
- to arrange to get a correctly formatted telex out of any text, even if the result might be something not really expected by the originator of the E-mail message
- to provide something (a program, a command procedure, an editor macro or whatever) on the local system used to send the telex by E-mail, which would

allow the user to know whether his telex is correctly formatted or not. Even in this case, a mechanism to handle badly formatted telexes should be implemented on the gateway, since the use of such a program, procedure or macro, could not be easily enforced.

The last problem, for the moment, but certainly not the least, is the one concerning the report messages, allowing the sender of a telex to be notified of the results of the operation, i.e. the success or the failure of the operation, with the cause of the latter.

The E-mail to Telex gateway as described here operates only in the direction E-mail to Telex, but it is planned to implement another gateway in the other direction as well if the need arises, as soon as the first one will be operating satisfactorily. However, it is quite clear that the implementation of such a gateway is not a trivial task and will be harder than for the first one.

6.10 EMDIR, CERN's directory service

To help people find the E-mail addresses of persons they want to send a message to, a directory service named EMDIR was introduced at CERN in 1987. Right now, the data base supporting the directory service is intended to contain only the addresses of people working on the CERN site. In the longer term, it should be possible to provide the same service to the whole HEP community as well.

EMDIR is implemented as an Oracle relational database located on one of the VAXes on the site. Each entry is composed of a name, a surname, a nationality, an institute or a CERN division, a telephone number, an office location, miscellaneous information and one or more E-mail addresses.

The database is accessed interactively from the major computers at CERN via remote procedure call over UDP/IP and DECNET (using Ethernet). On each of these hosts, a user interface to EMDIR allows the users to update their own entry. It is also possible to ask for the list of all entries corresponding to a given value for one or more of the attributes.

For those who do not work at CERN, it is possible, from any EARN/BITNET node, to send such a request using the TELL facility (interactive messaging). A service machine at CERN will answer using the same facility.

6.11 Transition to X.400

From the beginning of the gateway system implementation, it has always been very clear that gateways were only a short term solution, the final goal being the realization of a unique X.400 network linking everybody.

The "intercept strategy" followed by CERN had to allow the rapid interconnection of existing E-mail systems on different networks while at the same time it had to be flexible enough to integrate additional E-mail systems if necessary. Moreover, this strategy had to allow a fast and smooth transition towards X.400 by the different E-mail systems, and this at each's own pace.

The choice of X.400 as the mail protocol on which the strategy would be based was not the only possible one. For example, RFC 822 could also have been chosen, since it was (and still is) widely used. In particular, RFC 822 was then the most important mail protocol used on EARN/BITNET, which is one of the most widely used networks in the HEP community. This is currently changing since EARN is planning to migrate towards X.400 protocols, while BITNET plans to continue to use RFC 822.

However at the time of the study, X.400 was found to have the following advantages [Beyschlag85a] :

"- most PTT's will soon offer X.400 services, especially the integration of X.400 with Telex and Teletex
- most computer manufacturers will soon offer X.400 products, either complete X.400 based E-mail systems or just gateways between X.400 and their own products. Moreover, the 12 largest European computer manufacturers have declared that they are committed to implement the X.400 series of recommendations
- most if not all national research networks will base their message handling services on X.400
- X.400 is the first international standard that conforms to the OSI Reference Model and reaches up to the application layer, which has the advantage that communication is possible between computers of different manufacturers
- X.400 will lead to an international E-mail network of high functionality
- X.400 has been designed to be extendible to new technologies

- X.400 uses the store-and-forward technique and is restricted to layers 4 to 7 of the OSI model. This allows lower layers to be as heterogeneous as needed without affecting functionality
- the first X.400 implementations, EAN and COSAC, already exist".

There were also some arguments against the use of X.400 [Beyschlag85a]. Among these were the fact that X.400 was a new standard designed within an unusually short period of time, which could not avoid some ambiguities, leaving some issues subject to interpretation. Moreover, a number of points could not be completed and were left for further study. Another inconvenient was that the existing implementation of X.400 were not commercial products, and thus not as reliable as these and not having a comparable support.

However, X.400 was adopted and is still used at CERN as the standard towards which all E-mail systems are heading. The use of such a standard protocol for mail transfer is expected to offer increased connectivity, greater functionality and a reduced maintenance overhead [Heagerty87b].

Furthermore, this choice appeared later to be the good one since the three major networks to which CERN is connected (EARN/BITNET, EUNET/UUCP, DECNET) have also clear X.400 migration plans. CERN is even currently testing new X.400 products for IBM/VM on the one hand and for DEC machines on the other hand (DEC MRX in conjunction with the ALL-IN-1 package). X.400 implementations are also expected from the Norsk Data manufacturer.

CERN plans now to operate a gateway conforming to RFC 987 (the ARPA Internet standard for translation between X.400 and RFC 822). This was not necessary so far because EAN accepts addresses in RFC 822 format. It doesn't use the O/R name attributes like firstname, surname, ADMD, country and the like to code addresses, but rather domain defined attributes which allow an easy mapping to RFC 822 style addresses. In the future, such a gateway will offer connectivity with systems using the full X.400 characteristics. [Heagerty87b]

Chapter 7
A particular problem of a gateway manager : the analysis of statistics

In addition to the fact that gateway managers have to ensure the proper functioning of their E-mail gateways, at the technical level, they also have to take other factors into account. Some of these have more to do with administrative than technical issues, like the knowledge of most referenced sites, the accounting or the checking for illegal traffic. All these pieces of information allow the gateway managers to fine-tune their gateways and to operate them cost-effectively.

One of the means at the disposal of gateway managers to tackle all these issues is to analyze the log files produced during the activity of their gateways. However, log files contain only a huge amount of raw information that has to be processed somehow before becoming useful.

7.1 The implementation of a dedicated analysis tool

In order to draw meaningful data from log files, some specialized tool has to be designed to analyze the raw data and produce summaries and statistics, to derive other results from built-in formulas, to outline the evolution in time of some important figures, to check for illegal traffic, etc.

This section will introduce the description of the implementation of a such a software tool dedicated to the analysis of log files, and implemented by the author during his traineeship at CERN. Further details concerning the implementation will be discussed in section 7.4.

To create a software tool able to analyze log files, two approaches were possible. The first one was to write a (set of) program(s) in a high level programming language, like C or Pascal. The second one was to use a commercial package, namely SAS (Statistical Analysis System), dedicated to the production of all kinds of statistics from all kinds of data files. The package provides a language comparable to a high level language in many respects (declaration of variables, types, assignments, control structures, macro

instructions, etc) with a library of very specialized and powerful statistical functions (see [SAS85] for a detailed description of the language).

Both approaches have their pro's and con's. The major advantage of the high level language solution is the flexibility. Nearly everything that is needed can be programmed, even if that was not foreseen at the beginning. This is not the case with the statistical language, SAS, whose specialized functions can only process the main data structure called data set. Moreover, input/output operations can only treat these data sets too.

However, the SAS language was used, because of the availability of the statistical functions, which would have been difficult and tedious to program from the start, and also because of the proven robustness of the package. To cope with the inherent lack of flexibility of such a package, some "tricks" had to be used, mainly to allow an easy parameterization of the whole application (this was done by the inclusion in the main SAS program of SAS source files which were automatically produced by editor macros from some kind of configuration data files).

7.2 Data analyzed : the EAN log files

Following the strategy chosen to implement an E-mail gateway system at CERN (see section 6.3), there are two points were log files are produced. This means that all messages passing from one of the four E-mail systems that CERN interconnects to another one will be logged at one of these two points (some messages will even be logged twice). The first one is on the central gateway machine (MINT), and is more precisely constituted by the log file produced by sendmail on that machine. The second one is located on VXGIFT, the machine providing the entry/exit point to and from non-local hosts on DECNET and EAN, and is produced by the EAN software.

It was decided to start with the analysis of the EAN log files, which contains a line of information for each message exchanged at CERN between EAN and other E-mail systems or for EAN messages exchanged between CERN and non-local EAN hosts. Each line specifies the date and time the message was processed, a number indicating the logical channel which was used, the number of bytes the message contained, the type of the message, the message identificator, the address of the origin and an address for each of the recipients.

The type of the message can be one of the following :

- import : the message was imported from a neighbour MTA
- export : the message was exported to a neighbour MTA
- submit : the message was submitted by a local UA
- deliver : the message was delivered to a local UA
- import report : the imported message is a confirmation report
- export report : the exported message is a confirmation report

An typical line of an EAN log file would look like :

```
05/24/88 16:09:36 3 167 export <8805240542.AA01322@mint.cern>:
   <userX@hostA.CERN> <userY@hostB.EARN>
```

From all this, it should be clear that any message is normally logged twice in the file, since it is logged when it enters the EAN MTA (with type import, import report or submit) and when it is goes out of it (with type export, export report or deliver). This is why only those messages having as type export, export report or deliver were taken into account to build statistics.

The analysis of the EAN log files is just a first step, because it provides only partial results since messages exchanged between E-mail systems other than EAN are not taken into account. It is very important not to forget this fact when trying to interpret the results given by the analysis tool. Indeed, it would be very easy to draw grossly erroneous conclusions if only the EAN log file was considered.

7.3 Use of the log files

This section brings some details on what kind of results are produced by the analysis program from the log files, after the raw information has been analyzed. The examples given are inspired from real data.

7.3.1 Knowledge of most referenced hosts

As already stated, one of the precious information that can be found in log files is the names of the hosts which are the most referenced, i.e appearing either as origin or

destination of messages. The search for most referenced hosts is done for all the hosts without distinction on the one hand, and only for CERN hosts on the other hand.

This information can help a gateway manager decide what kind of lines to set up between CERN and the hosts that are regularly called (high speed - high cost lines for very popular hosts, cheaper and slower lines for others). New lines may also be established directly to hosts with which lots of data is regularly exchanged, with no need to pass through intermediate relay hosts.

20 most popular originating hosts

OBS	HOST_ORG	DOM_ORIG	OBS_NUMB
1	VXCERN	DECNET	881
2	CERNVM	CERN	419
3	VAX	UNINETT	299
4	CERNVAX	CERN	149
.	...	...	...

Fig 7.1 : Most referenced hosts (as origin)

Figure 7.1 shows how that information is presented to the gateway manager. In the figure, OBS stands for observation number, then HOST_ORG indicates the originating host. Likewise, DOM_ORIG indicates the originating domain, and last, the column OBS_NUMB shows the number of messages (or observations in SAS terms) which were sent.

The name and domain of the hosts, either local or external to CERN, appear in order of importance, depending on the number of messages that they sent. The information concerning the hosts who received the most messages is shown in another list (i.e "Most referenced hosts (as destination)").

From these figures, VXCERN (which is the central VAX/VMS cluster at CERN) appears to be, at least for the period analyzed, the host having sent the greatest number of messages (to be precise, 881 messages). It is followed by the central IBM machine (CERNVM), and only then comes a host external to CERN into play (VAX.UNINETT).

This kind of information is also of special interest when CERN hosts only are taken into account, since it then indicates which CERN hosts exchange the most messages with the outside. Figure 7.2 is an example of the results one could find.

Number of messages received, sent and exchanged by .CERN hosts

OBS	HOST	RECEIVED	SENT	EXCHANGED
1	CERNVM	452	419	871
3	CERNVAX	111	149	260
4	PRIAM	157	74	231
5	VXGIFT	82	136	218
6	GEN	69	59	128
.	...	...	...	...

Fig 7.2 : Most popular CERN hosts

(NB : CERNVM appears as a host belonging to domain CERN, even though it is an EARN host. This is because, as can be seen in page 19 of Appendix 2, the remapping facility of the analysis program was used to force the domain of some CERN hosts to be "CERN", in order to facilitate the detection between local hosts and others.)

7.3.2 Knowledge of most referenced domains

Gateway managers like to know which are the most popular domains, either as origin or as destination of messages. Since we talk about the EAN log files, domains mean the management domains into which the EAN E-mail system is divided. They comprise pure EAN/X.400 domains (like CERN for all the EAN hosts at CERN, DE for Germany, NL for Netherlands, FR for France, etc) but also domains which have been artificially set up to allow the EAN MTA to recognize the messages that have to be gatewayed to other E-mail systems (like BITNET for EARN/BITNET hosts, UUCP for EUNET/UUCP, DECNET for DECNET, and EDU, COM and GOV for ARPA Internet).

A list of these domains in decreasing order of popularity may give useful information. For example, are the EAN/X.400 domains more popular today than yesterday, or has the X.400 standard less impact as originally thought ?

From figure 7.3, it can be deduced that for the analyzed period (and generally, if the period is long enough) most messages passing through the EAN MTA whose log file is analyzed come from the CERN site. This is quite normal since the gateway service has originally been set up as a local service, and that it is used by external sites only on the basis of a special agreement with CERN.

Closely following come then DECNET and EARN/BITNET, which are the two networks, as the figures confirm, most used for long distance communications in the HEP community.

Origin domains importance

OBS	DOM_ORIG	OBS_NUMB
1	CERN	1315
2	DECNET	1032
3	EARN/BITNET	848
4	UNINETT	444
5	CH	311
.	...	...

Fig 7.3 : Most referenced domains (as origin)

UUCP is indeed currently little used for scientific exchanges in general, in the HEP community in particular. However, once again, it must not be forgotten that this log file don't take into account all the messages passing through CERN. In the particular case of UUCP, only those messages exchanged between EAN and UUCP and between DECNET and UUCP appear in the log file analyzed. The traffic between other networks (including EARN/BITNET) and UUCP, as well as the traffic between UUCP hosts, is logged in the sendmail file on another machine.

7.3.3 Summary of traffic between domains

The list of most referenced domains only indicates the number of messages sent to or from the most popular domains. But it would also be interesting to know between which domains messages are exchanged. This information is summarized in a matrix showing for each domain, the number of messages it has sent to and received from all other domains.

Figure 7.4 shows that matrix as it is produced by the analysis tool. Note that the last column (i.e. "Total") should contain the same figures as those presented in fig 7.3 (most referenced domains as origin), except that they are not sorted by decreasing order of importance. Likewise, the last line should contain the same figures as the ones presented in the list of the most referenced domains as destination (not shown here). The lower right corner indicates the total number of messages that were logged for the period under analysis.

Summary of traffic between all domains (short form)

Number of messages / Origin	Destination Domains						
	ARPA	CDN	CERN	CH	DECNET	...	Total
CDN	.	.	18	.	4	...	22
CERN	1	46	527	55	341	...	1315
CH	.	.	20	2	1	...	311
COM	.	.	1	5	.	...	16
DE	.	.	18	.	13	...	31
DECNET	1	4	516	.	35	...	1032
...	...	...	...	...	...	...	...
Total	2	52	1360	339	721	...	4364

Fig 7.4 : Summary of traffic between all domains (short form)

The dots in the matrix indicate that no message was exchanged between the corresponding domains. For example, it would be abnormal to find a positive figure in the element indicating the number of messages exchanged between CDN and ARPA, because on the one hand, there is direct gateway between CDN (the Canadian EAN domain) and ARPA (the US ARPANET), and on other hand, such messages would be illegal, because neither CDN or ARPANET have special agreements with CERN to use their gateways.

7.3.4 Basis for accounting

Accounting can be seen from two different view points. On the one hand, it has to do with the amount of money one has to pay for the use of some service, in this case the E-mail service, and on the other hand, it can also be used by the providers of a given service, in this case gateway managers, to know how much it costs them to operate such a service.

Only the second point of view will be taken into account here. There are several reasons for this. First, the gateway service is provided freely to all local users , because CERN is funded centrally, so that there is no need for individual bills for users or divisions. This could change in the future, each division having then to pay for its own usage of the gateways. Second, the gateway service provided at CERN is not very old (less than two years of actual usage). In the beginning, only the most urgent and important things had to be coped with. Third, the cost incurred for the transmission of data between CERN and other sites is currently taken into account by

other means. For example, some sites have agreed to poll CERN regularly, thus paying for the costs of transmission, others allow CERN to poll them with reverse charging. However, these costs do not take into account the people, hardware and software necessary to operate the gateway service correctly.

Another reason why the accounting will only have to do with the managers' point of view is that until recently, there was no means at all to know more or less precisely what was the throughput of the gateways, which were the hosts most often called and most of all, what it is costing CERN to operate the gateway service for others (i.e. non local hosts). The only thing gateway managers could do was to grossly estimate these figures. A first step towards a better knowledge of these figures has been done with the implementation of the analysis program described in this chapter.

Thus, one of the most important things to know when operating a service involving external communications is the cost of these. Since it is rather difficult to calculate the actual figures, it was decided to assess them as precisely as possible. For each message, the formula used takes into account the size (in number of bytes) and the destination, since only these factors vary in the calculation of the transmission cost of any message.

To take into account the fact that some headers and trailers are added the to data to be transmitted by the transport mechanism, the number of bytes is multiplied by some constant. The fact that there are errors on the communication links and that some messages have to be repeated is also taken into account in that constant.

Then, from the destination domain of the message, the cost of transport per volume is found in a parameters file, and is multiplied by the actual number of bytes to transmit. The cost per transmission time is then added, in function of the number of bytes to transmit, and the result is an acceptable estimation of the actual cost of the sending of the message (more details on the way to compute this result can be found in Appendix 2, p. 20).

The costs related to messages having the same domains as destination are added and presented in an extended form of the matrix mentioned in the previous section. The extended form of that matrix will contain, in each place already indicating the number of messages exchanged between the corresponding domains, the additional following information :

- the percentage the number of exchanged messages represents with respect to the total number of messages sent

- the total number of bytes contained in the messages sent between the two domains involved
- the mean of the number of bytes per message
- the costs (in Swiss franks in CERN's case) incurred to sent all the messages between the two domains involved.

An extract of such a matrix is shown in figure 7.5. Note that the actual figures are truncated to the lower integer, which explains the zero's in the Percent and Cost columns.

Summary of traffic between all domains (extended form)

Number of messages / Number of	Destination Domains							
	...	DECNET						...
Origin	...	Nb msg	Percent	Bytes	B Mean	Cost SF		...
CDN	...	4	0	5246	1312	0		...
CERN	...	341	8	1207539	3541	83		...
CH	...	1	0	613	613	0		...
COM	...	.	.	.	.	.		...
DE	...	13	0	158681	12206	11		...
DECNET	...	35	1	219625	6275	15		...
...	...	...	...	...	...	...	...	...
Total	...	721	17	2519543	—	174		...

Fig 7.5 : Summary of traffic between all domains (extended form)

It is also worth noting that in two cases, the transmission of messages doesn't cost anything to CERN. The first case is when the message has a CERN host as destination, since then it is transferred using dedicated local lines (e.g. using TCP/IP or DECnet over Ethernet). The second case is when the message is destined for a domain which has agreed to pay for the transmission. In that case, the costs appear as negative in the results, which indicates to the gateway managers, not what the transmission of messages to these domains costs, but rather what they save by not paying for these transmissions. This is also the case for messages exchanged with EARN/BITNET hosts since the costs are constant, because all the links have to be leased lines.

7.3.5 Summary of traffic between superdomains

The matrices mentioned in the two previous sections are very useful when detailed information is needed, but sometimes, gateway managers would like to catch the most important figures at first glance. This is the reason why another matrix is produced, where all the EAN/X.400 domains are gathered under one single superdomain, as well as the ARPA Internet domains are gathered under another single superdomain.

Summary of traffic between superdomains (short form)

Number of messages	Destination Superdomains						
Origin	ARPA/ Internet	CERN	DECNET	EAN/ X.400	EARN/ BITNET	...	Total
ARPA/Inter	.	5	1	28	.	...	46
CERN	4	527	341	251	111	...	1315
DECNET	39	516	35	23	335	...	1032
EAN/X.400	.	143	19	8	572	...	852
EARN/BIT	.	101	245	490	1	...	863
UK	.	48	44	26	6	...	128
UUCP	.	20	36	63	5	...	128
Total	43	1360	721	889	1030	...	4364

Fig 7.6 : Summary of traffic between superdomains (short form)

In that matrix (shown in fig 7.6), there remain only seven superdomains, namely ARPA Internet, CERN, DECNET, EAN/X.400, EARN/BITNET, UK and UUCP. The "extended" version of this matrix, in the same sense as in the previous section, is also produced.

7.3.6 Detection of illegal traffic

As already stated, the gateway service provided by CERN is not a public one. Since it costs money to operate such a service, and also because of the legislation concerning the switching of third-party traffic, only those sites having an explicit authorization from CERN can use the gateway service to send messages to a non-CERN site.

It is not easy to automatically prevent non-authorized users to use the gateways. However, it is not difficult to check in the log files that each message was authorized (thanks to a configuration file containing the names of those domains having special agreements with CERN).

Thus, illegal messages are brought to the attention of gateway managers which take further action if necessary. This information is presented as a list composed of one line for each detected illegal message. Each line contains the date and time the message was processed, its type, its size, the address of the user who sent it and the address of the recipient(s).

7.3.7 Evolution of the load

One of the most important figures that can be drawn from the log files is the total number of messages that has passed through the gateway during a given period. From this figure, the mean number of messages per day can be easily calculated. This value can in turn be compared to the values that were computed for the previous periods.

These figures taken together give a good idea of the evolution of the gateway service as a whole, and allow gateway managers to assess the future global trend and to foresee the decisions that will have to be taken.

Figure 7.7 shows how these figures will eventually appear. The analysis program being only used since recently, actual figures are not precisely known, so that the example may not be very accurate or even consistent.

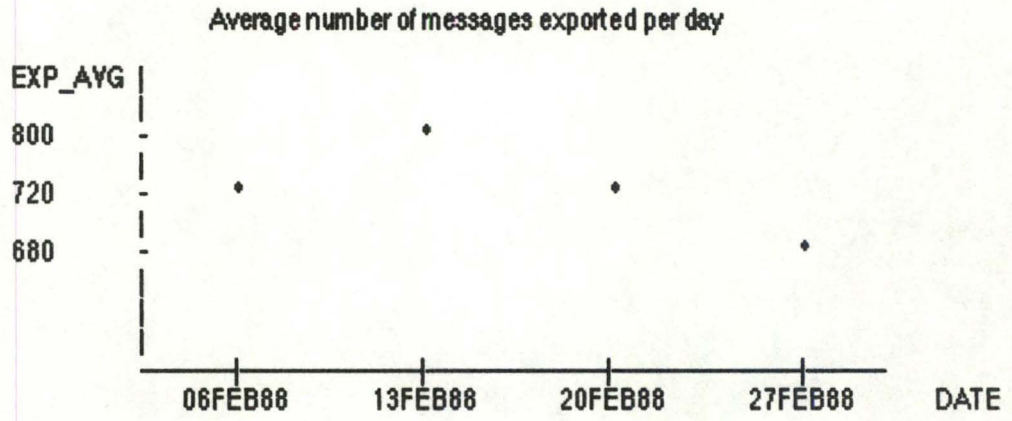


Fig 7.7 : Evolution of global traffic

The graphical representation is very useful and allows gateway managers to grasp in one glance the evolution of the traffic logged in the analyzed log file. In the example,

one can see that the analysis job was started (or restarted after initialization) on the 6th of February, and that it has run (automatically) each week from then on.

(Note that SAS manages carefully the graphical representation so that the space between the observations is optimized in function of the space available. This means that the global evolution of the traffic during one year will be as easily catch at first glance as the evolution during one single month.)

For those who like accurate figures, the evolution of the load is also presented as a list of dates (at which the analysis was done) with the number of messages logged during the corresponding period and the mean number of messages logged per day during that period.

7.4 Some details about the implementation of the analysis tool

The analysis tool producing the results described in the previous section from EAN log files was developed in the IBM VM/CMS environment, and was based around the SAS statistical package. It is composed of SAS programs, indicating how to parse and analyze the input files, of command files which control the sequencing of the whole application, and of text files containing the data used to parameterize the SAS programs and the command files.

The structure of the main SAS program will first be described, and then the automatization of the whole analysis process will be explained.

7.4.1 Structure of the application

The task of the main SAS program is twofold. On the one hand, it has to parse an input file, in this case the EAN log file, and to produce the result described above. On the other hand, it must also be possible to accumulate the new results with the ones calculated previously, so that it becomes possible to have results available for the last week, the last month or even the last year (the value of these periods can be parameterized). The results of the analysis are of course kept in a compacted form, otherwise they would be too bulky and impossible to manage, because of the amount of data which is logged daily.

Every day, the EAN log file produced on the VXGIFT machine is transferred on CERNVM, the central IBM machine, where it will stay until it has been analyzed by the SAS program. This transfer happens automatically, using a product called INTERLINK, which allows file transfer between VAX/VMS and IBM/VM machines.

Under the control of a command file (see next section), the SAS program is called to analyze the log files. One important point should be noted here : the SAS program is decomposed in modules, each having a specialized task. One of them reads the input file and put it in a standard internal format (data set). Another one takes as input the data set produced that way and produces a summarized version of it. Another one adds the current data set with data sets containing the results of previous periods. Yet another one produces from a data set the results in a printed form.

One of the direct benefits of this "modularization" is that when the whole analysis job will be stable enough, it will be possible to modify it slightly to be able to analyze the log files produced at the other gateway than EAN, namely the log files coming from sendmail on the central gateway machine.

Indeed, as already stated, all log files contain more or less the same information about each processed message : date and time, size, origin, destination and type of processing. So that the only module that will have to be rewritten to reuse the whole application will be the module reading an input file and producing a corresponding data set in standard format. From then on, all the other operations are the same for every new type of log file : summary, calculation of statistics, accumulation with previous results, production of printed results, automatization of the sequencing of the whole application, etc.

Another point which is also important is that the SAS program is parameterized with files containing every type of parameters :

- the names of the domains likely to appear in addresses, and for each of these domains, its corresponding superdomain, country and indication whether it costs something to send messages there or not
- the costs per volume associated with each possible destination country
- the formula and the parameters used to calculate the cost per message
- the remapping of the names of some hosts because some (CERN) hosts are known under several names
- the names of the CERN hosts whose domain has to be forced to "CERN" to point out that they are local

- the remapping of the names of some domains, because some domains have several names (e.g. CH and CHUNET, OZ and AU, DFN and DE, DNET and DECNET, etc)
- the names of domains having explicit authorization to use CERN's gateways
- miscellaneous parameters as the number of most popular hosts to print in the results

All these files contain the value of the parameters in plain text. This means that the files are easily read and understood by a human reader and that they may be modified using plain text editors. These files are then processed, automatically, by editor macros to be transformed in a form compatible with SAS programs.

7.4.2 Automation of the process

In order for the gateway manager not to support the burden of gathering the input files, launching the analysis program at regular intervals, and also to relieve him of the chore of having to know every little detail about how the different programs fit together, a batch job was written in REXX, the command language used in the IBM VM/CMS environment.

Each time it wakes up, the batch job calls a routine which will take the input file (composed of the EAN log files not yet analyzed) and ask the SAS program to analyze it and add the data produced (in a compressed form) to the SAS data sets containing the data produced during the previous analyses. The number of days to wait before the batch job wakes up is a parameter easily modified.

The batch job produces a printed report of the data it has accumulated so far when it's time to do so. It knows the right time has come by checking the number of times it has waken up since the last printing of results, against a given parameter. This parameter tells the SAS program how many times to simply analyze the data and add it to the current SAS data sets before producing (and printing) a report.

At the end of the execution of the batch job, the latter resubmits itself to automatically wake up later, at some given time.

If anything goes wrong during the execution of the batch job, a message is automatically sent via E-mail to the gateway manager. Some of the problems likely to happen are a temporary lack of memory (the amount of data to analyze is huge and requires a lot of working space), the impossibility to access the input file, the lack of

enough space on the disk. When the results have been successfully printed, a message is sent to the gateway manager to tell him his statistics are available and ready to be read by an experienced eye.

7.4.3 Personal opinion

The gateway service operated at CERN is recent. The COMICS study ended in 1985 and the gateway system became officially operational in March 1986. The initial implementation effort was approximately one person-year, and the continued maintenance of the gateways consumes the equivalent of one full-time person-year [Heagerty87b]. The technical problems have been and are still numerous, and the human resources are limited.

All this explains that until now, gateway managers haven't had to opportunity to develop an oppropriate tool to assess more or less precisely the use of their gateways, the cost to operate them, who uses the gateway service, etc.

The analysis tool described in this chapter is a first step in the systematic analysis of all these questions, from the log files produced during the activity of the gateways. The first results, even though very simple, seemed nevertheless quite interesting and sometimes even unexpected for the gateway manager who asked for the development of such an analysis program.

However, the analysis tool described is only a first step. First, it will have to be run for a certain period of time before becoming stable. During that period, some errors will be corrected, new functionalities will be added and likewise, existing functions will be removed. The parameters of the whole process will also have to be fine-tuned (e.g. intervals of time between two successive analyses, more technical parameters like the amount of work space needed for the SAS program to proceed properly, etc).

After that first period during which the analysis job will try to prove it is worth the effort needed to maintain and update it, and when it will be stable enough, it will be adapted to the analysis of the other log file produced by the gateway system, i.e. the log file of sendmail on the central mail exchange machine. When the results comparable to the current ones will be available for the second log file, very precise and very interesting conclusions will be drawn.

For the adaptation in question, only a well defined and limited part of the main analysis program will have to be modified. However, the effort required should not be

underestimated, mainly because of the variety of address formats sendmail recognizes and logs.

The difficulties encountered by the author during the development of this application are threefold. First, the programming environment was new. the IBM VM/CMS environment is not really comparable to the one provided by other common operating systems, like Unix or VMS. However, when well understood, this environment proves to be very powerful. Likewise, the REXX programming language is very powerful and flexible.

Second, the statistics package, SAS, had to be learned. It provides a high level programming language, very powerful for all that has to do with statistics, but can be sometimes clumsy and not flexible enough.

Third, the log files themselves are bulky and thus difficult to handle. Moreover, they have internal structures which are not always very obvious. Moreover, the address formats that can be found in the EAN log files in particular are numerous and sometimes very unexpected. Indeed, the address format that one would expect in such a log file is the well known RFC 822 address format :

```
user@host.domain
```

Most of the addresses fit in this pattern. However, here are some examples of addresses that were really found in the analyzed log files :

```
user@uk.ac.bham.ph.g%rl.ib
user.cernvm.cern
39992::user.decnet
user%HTIKHT5.bitnet
P136@QZCOM.bi tnet
g05a01`frccsc21.earn
cernvax!ethz!user.uucp
KERMSRV@CUVMA.BITNET.IBM-PC.MS-DOS.Kermit.Y2.29.Terminal.Emulator...
    (continued)       ...Summary.Keyboard.Layout.and.Escape.Sequence....
user at cern.earn
user:sbcchail.bitnet
```

A program like sendmail is happy to process such addresses because it has been written with that goal in mind, and has all it needs to do so (e.g. a configuration file with complex rewriting rules that can be updated at will). But for an analysis program written from scratch, it is not so easy to find the hosts and domains corresponding to such exotic addresses.

Conclusion

Nowadays, computers are more and more used for a variety of information processing tasks. We have already entered the post-industrial period, the age of information. In the information society, new communication means between men will be more and more needed. One of these modern ways to communicate will surely be the use of electronic mail, faster, more reliable, more functional, more flexible than traditional mail, and yet cheaper. Electronic mail will be increasingly used not only by scientists and business men, but also by common people, for work or personal usage.

However, one of the greatest dangers threatening the promising future of electronic mail is the incompatibility. Indeed, there exist currently a bunch of different electronic mail systems, speaking different protocols and offering different functionalities.

This problem of "incommunicability" has been partly solved by setting up electronic mail gateways. The latter are used to fill the gaps between incompatible electronic mail systems, to interconnect them and to ensure the minimum loss of functionality. Several techniques may be used, but none of them is entirely satisfactory. Each brings its specific problems and limitations and adds them to the difficulties inherent to electronic mail systems.

Gateway managers do their best to cope with all these problems, but the difficulties are not only technical. Administrative issues come into play as well, and sometimes require the collaboration of decision-making bodies at an international level, for example to solve the problems of name clashes.

Another important task of gateway managers is to ease the use of their gateways. Indeed, the confusion surrounding the topic of how to address people is a serious deterrent for newcomers to electronic mail services.

It is quite clear now that incompatibility is the main plague threatening electronic mail and that this threat is real. So, is electronic mail doomed to the same lot as the Tower of Babel ?

No, certainly not. First, the need for that kind of service is obvious and pressing. Second, as often in the computer science history, the key word will prove to be "standard".

Indeed, if an electronic mail standard was used, most problems would disappear automatically. There would be a worldwide connectivity, everybody in the world being able to send messages to each other. There would be no more loss of functionality incurred by crossing gateways. The addressing problems would be wiped out since one single syntax would be used by everyone. The problem of name clashes would be seriously decreased.

Such a standard exists. In fact, the problem is that there are even two of these. RFC 822, supported by the US Department of Defense, rules electronic mail in most of the USA and in a good part of the rest of the world. It is a de facto standard widely used. The CCITT has also released its standard, an official international one, X.400. Europe has definitely decided for the adoption of X.400 and experiments are under good way. Furthermore, most manufacturers offer or will soon offer X.400 products.

The RFC 822 world is also expected to eventually adopt X.400. The transition from current mail protocols to X.400 will not occur overnight, of course, but the prospects look good. However, the migration to X.400 appears a necessary step if electronic mail is to become one of the major communication means of the future.

Bibliography

[Allman83] Eric Allman, Sendmail, an internetwork mail router, University California Berkeley, July 1983

[Beyschlag85a] Ulf Beyschlag, COMICS final report, CERN/DD, Feb 1985

[Beyschlag85b] Ulf Beyschlag, "Going X.400, Example of an intercept strategy", Second International Symposium on Computer Message Systems, Sept 1985, pp. 27-39

[Bryant87] Paul Bryant, "The migration of EARN to use ISO protocols", Computer Networks and ISDN Systems 13-3, 1987, pp.201-203

[CCITT84] CCITT Study Group VII, X.400 Series Recommendations, June 1984

[Crocker82] David Crocker, Standard for ARPA Internet Text Messages, RFC 822, Aug 1982

[Crosswell82] Alan Crosswell, Batch Simple Mail Protocol, Sept 1982

[Davies83] Julian Davies and Reg Quinton, MLNET, A general purpose message gateway architecture, University of Western Ontario, 1983

[Heagerty86] Denise Heagerty, Electronic mail status, CERN/DD, Sept 1986

[Heagerty87a] Denise Heagerty, "Pratical experience with high level gateways for mail transfer", Computer Networks and ISDN Systems 13-3, 1987, pp.195-200

[Heagerty87b] Denise Heagerty, Experience with mail gateways and transition to X.400, CERN/DD, 1987

[Heagerty87c] Denise Heagerty, A user guide to electronic mail services at CERN, CERN/DD/US, Jan 1987

[Heagerty87d] Denise Heagerty, Proposal for a store and forward DECnet gateway, CERN/DD, Oct 1987

[Henken87] Gerrit Henken, "Mapping of X.400 and RFC 822 addresses", Computer Networks and ISDN Systems 13-3, 1987, pp.161-164

[Horton86] Mark Horton, UUCP Mail Interchange Format Standard, Bell Laboratories, 1986

[Kaufmann87] Peter Kaufmann, "Migration of DFN Message Handling Service", Computer Networks and ISDN Systems 13-3, 1987, pp.207-211

[Kill86] Steve Kille, Mapping between X.400 and RFC 822, RFC 987, June 1986

[Landweber86] Lawrence Landweber, Dennis Jennings, Ira Fuchs, "Research computer networks and their interconnection", IEEE Communications 24-6, June 1986, pp.5-14

[Magedanz87] Thomas Magedanz, Michael Tschichholz and Karl-Heinz Weiss, Interconnection of X.400 Systems and notable gateways, IFIP proceedings, 1987, pp.1.1-1 - 1.1-19

[Meyer84] Theodore Meyer, The MHS recommendations, structure and implementation perspective, GTE Telenet Communications Corporation, 1984

[Neufeld85] Gerard Neufeld, The EAN distributed message system (User's manual), University of British Columbia, Feb 1985

[Neufeld87] Gerard Neufeld, The EAN distributed message system (Administrator's guide), University of British Columbia, July 1987

[Nowitz78] D. A. Nowitz, M. E. Lesk, "A dial-up network of Unix Systems", Unix documentation, Aug 1978

[Quaterman86] John Quarterman and Josiah Hoskins, "Notable computer networks", Communications of the ACM 29-10, Oct 1986, pp. 932-971

[Postel82] Jonathan Postel, Simple Mail Transfer Protocol, RFC 821, Aug 1982

[Redell83] David Redell and James White, Interconnecting Electronic Mail Systems, Computer, Sept 1983, pp.55-63

[SAS85] SAS User's Guide : Statistics, SAS Institute Inc., 1985

[Schicker84] P. Schicker, An Introduction to the CCITT X.400 series recommendations, Appendix 1 to the CCITT X.400 series recommendations, 1984, pp.A.1-1 - A.1-10

Appendix 1 : List of abbreviations

ADMD	ADministration Management Domain
ARPA	Advanced Research Projects Agency
ARPANET	Advanced Research Projects Agency NETwork
BITNET	Because It's Time NETwork
BITNIC	BITnet Network Information Center
BSMTP	Batch Simple Mail Transfer Protocol
CCITT	Comité Consultatif International des Télégraphes et Téléphones
CERN	Organisation Européenne pour la Recherche Nucléaire
COMICS	COmputer based Message systems InterConnect Strategy
COSAC	COmmunication SAns Connection
DEC	Digital Equipment Corporation
DFN	Deutches ForschungsNetz
DIF	Document Interchange Format
E-mail	Electronic mail
EAN	Electronic Access Network
EARN	European Academic Research Network
ERN	Entry Relay Node
FTP	File Transfer Protocol
GMD	Gesellschaft fur Mathematik und Datenverarbeitung
HEP	High Energy Physics
IBM	International Business Machines
IFIP	International Federation for Information Processing
INRIA	Institut National de Recherche en Informatique et en Automatique
IPMS	InterPersonal Message System
ISO	International Standards Organization
JANET	Joint Academic NETwork
JNT	Joint Network Team
MHS	Message Handling System
MLNET	Maple Leaf NETwork
MTA	Message Transfer Agent
MTAE	Message Transfer Agent Entity
MTL	Message Transfer Layer
MTS	Message Transfer System
O/R	Originator / Recipient
OSI	Open Systems Interconnection
PRMD	PRivate Management Domain

RARE	Réseaux Associés pour la Recherche Européenne
RFC	Request For Comments
RSCS	Remote Spooling Communication Subsystem
RTS	Reliable Transfer Service
SDE	Submission and Delivery Entity
SMTP	Simple Mail Transfer Protocol
SPAN	Space Physics Analysis Network
TCP/IP	Transmission Control Protocol / Internet Protocol
TCx	Transport Protocol x (x = 0..4)
UA	User Agent
UAE	User Agent Entity
UAL	User Agent Layer
UBC	University of British Columbia
USENET	USEnix NETwork
UUCP	Unix to Unix CoPy

Appendix 2 : Dedicated analysis tool source listing

This appendix contains the listing of the source code of the set of programs implementing the analysis tool described in chapter 7. They were written by the author during his six month traineeship at CERN in the second semester of 1987.

Some of the programs are written using the SAS statistical language. These are responsible for the actual analysis of the log files. The other programs are rather command files written in REXX, the command language used in the IBM VM/CMS environment.

The command files control the sequence of the operations. They ensure that all the resources needed for a proper execution of the SAS programs (i.e. input file, disk space, memory space, etc) are available, they provide general commands to launch and stop the processing, ensure the automatic execution of the analysis process at regular intervals and generally speaking, they relieve the gateway manager of the burden of having to know every little detail of the way it all works.

A listing of the files used to parameterize the application is also given. These files are plain text files that are automatically processed by editor macro's before being integrated to SAS programs. They contain all the data that should be easily modified by gateway managers, like the names of the known hosts and domains, the mapping of names for some hosts and domains, the formula to calculate the costs incurred to send messages, etc.

Last, a more detailed specification of the whole job is given, as well as a list of all the files constituting the analysis tool.

```
/*************************************************************************
* SASWEEK : This SAS program analyses EAN log files and add the infor *
*           mation to the current EAN data sets                         *
*************************************************************************/

 options pagesize=110 linesize=120;
*options pagesize=22  linesize=79;
 options replace;              /* to allow the replacement (or updating */
                               /* of the current data sets              */

/* definition of the external files needed */

CMS FILEDEF DATAFILE DISK EANFILE DATA D;   /* EAN log file */

CMS FILEDEF HOSTEST DISK HOST TEST E;       /* remap of host */
CMS FILEDEF DOMTEST DISK DOMAIN TEST E;     /* remap of domain */
CMS FILEDEF HTDMTEST DISK HOSTDOM TEST E;   /* map of domain from host */
CMS FILEDEF CERNTEST DISK NOTCERN TEST E;   /* removes .cern if necces.*/
CMS FILEDEF COUNTMAP DISK DOMCOUNT TEST E;  /* map of domain->country */
CMS FILEDEF CHKALLOW DISK ALLOWED TEST E;   /* check if msgs allowed */
CMS FILEDEF CONFIG DISK CONFIG TEST E;      /* gives some config data */


/*************************************************************************
* Construction of data set comprised of the raw data                    *
*************************************************************************/

/* We will build a SAS data set containing, for each message found in
   the input file, its date and time, its number of bytes, its type,
   its origin and destination address under original format, its origin
   and destination domain part only (part after the @ sign in the
   address that we will call host-domain pair) under normalised format
   We will also build a data set containing the date and time of the
   first and last message, and for later use, we will also  buld a data
   set containing the date and time of the first message, and one for the
   date and time of the last message.
   These four data sets are temporary, not permanent, since they are only
   a first step to the update of the existing data sets */

  ta raw_data  (keep = date time nbytes type orig dest hostdomo hostdomd)
     time_dat  (keep = event time date)
     date1     (keep = frstdate)          /* date and time of 1st msg  */
     date2     (keep = lastdate);         /* date and time of last msg */

  infile datafile eof=endlabel ;
  retain date time year nbytes channel type msgid orig
         hostdomo hostdomd frsttime lasttime frstdate lastdate;
  format date frstdate lastdate DATE7.;
  format time frsttime lasttime time8.;
  format orig orig2 dest dest2 msgid msgid2 hostdom $150.;
  format hostdomo hostdomd $70.;
  format type $10.;

  informat date frstdate lastdate DATE7.;
  informat time frsttime lasttime time8.;
  informat orig orig2 dest dest2 msgid msgid2 smth
           datestr hostdom $150.;
  informat hostdomo hostdomd $70.;
  informat type $10.;

  false = 0;
  true = 1;
  frstdate = missing;

  /* get current year in the config file */
  %include config;

  input datestr $ @;       /* input first date string */
  iter = 0;
  do while (true);
    iter = iter+1;
    input time time8. channel nbytes @;

    /* date processing */

    i = index(datestr,'/');
    day = input(substr(datestr,i+1,2),2.);
    month = input(substr(datestr,1,i-1),2.0);
    date = mdy(month,day,year);

    if frstdate = missing then do;
       frsttime = time;  frstdate = date;
       lasttime = time;  lastdate = date;
    end;

    if date <= frstdate then do;
       if date < frstdate then do;
             frstdate = date; frsttime = time;
       end;
       else if time < frsttime then frsttime = time;
     end;

    if date >= lastdate then do;
       if date > lastdate then do;
             lastdate = date; lasttime = time;
       end;
       else if time > lasttime then lasttime = time;
     end;

    /* get the type of the message */

    input type $ smth $ @;
    if smth = 'report' then do;
       type = trim(type)||'_rep';
       input smth $ @;
    end;

    /* get entire message ID (terminated by ':') */

    msgid = smth;
```

```
    do while (substr(msgid,length(msgid),1) ~= ':');
          input msgid2 $ @;
          msgid = trim(msgid)||' '||trim(msgid2);
    end;

    /* origin processing (terminated by '>') */

    input orig $ @;
    do while (substr(orig,length(orig),1) ~= '>');
          input orig2 $ @;
          orig = trim(orig)||' '||trim(orig2);
    end;

    addr = orig;
    link proc_adr;          /* get the host-domain part of the address */
                            /* (part after the @ sign) in a normalised */
                            /* format, with the part .CERN removed if  */
                            /* necessary                               */
    hostdomo = hostdom;

    /* if it's not a report message, then get the destination(s) part */

    if type='import' or type='export' or type='submit' or type='deliver'
    then do;

      /* destination(s) processing */

      input dest @;
      fini = false;
      do while (~fini);             /* do for each of the destinators */
        do while (substr(dest,length(dest),1) ~= '>');
            input dest2 $ @;
            dest = trim(dest)||' '||trim(dest2);
        end;

        addr = dest;
        link proc_adr;              /* get host domain pair (see above) */
        hostdomd = hostdom;

        output raw_data;            /* add the info for this message in */
                                    /* the correct data set             */

        /* are there other recipients for this message ? */

        input dest $ @;
        if substr(dest,1,1) ~= '<' then do;
            fini = true;
            datestr = dest;
        end;
      end;  /* do while (~fini) */
    end;  /* if type = ... then do */
    else do;                        /* this is a report message */
      output raw_data;
      input datestr $ @;
    end;
  end;  /* do while (true) */
  return;


 /* processing of an address : we first check if we know the format
    of the message (no '=' in the address (meaning X.400 style format),
    a '@' is required, and a '.' after the '@' sign).  If it's not known,
    we set the host-domain pair to 'UNKNOWN' which will have to effect
    that the info concerning the message will be output in a special data
    set containing unrecognized format messages and messages containing
    an unknown domain */

proc_adr:

  /* first we remove a possible '/C=country' from an hybrid format
     address */
  C_index = index(addr,'/C=');
  if C_index ~= 0 then addr = substr(addr,1,C_index-1) || '>';
  if index(addr,'=') = 0 then do;  /* not an X.400 address */
    hostdom = upcase(addr);
    at_pos = index(hostdom,'@');
    if at_pos ~= 0 then do;           /* is there an '@' sign ?     */
      do until (at_pos=0);            /* yes, then get address part */
                                      /* after the rightmost one    */
        hostdom = substr(hostdom,at_pos+1);
        at_pos = index(hostdom,'@');
      end;

      /* we know have our host-domain pair but is there at least one dot
         in it ? */
      dot_pos = index(hostdom,'.');
      if dot_pos ~= 0 then do;

        /* This will take out the superfluous .CERN part from the
           domain name when necessary */
        if index(hostdom,".CERN>") ~= 0 then do;
          %include cerntest;
        end;

        hostdom = substr(hostdom,1,length(hostdom)-1); /* remove '>' */

      end;  /* if dot_pos ~= 0 then */
      else hostdom = 'dummy.UNKNOWN';   /* sorry, there was not dot... */
    end;  /* if at_pos ~= 0 then */
  else hostdom = 'dummy.UNKNOWN';       /* sorry, there was not '@'... */
  end;  /* if index(addr,'=') = 0 then */
  else do;    /* seems to be a X.400 address */
    hostdom = 'dummy.UNKNOWN';
  end;
  return;

  /* only when all the observations in the input file have been read
     are we able to get the date and time of the first and last
     messages */

  endlabel: do;
    event = 'First message';
```

```
    time = frsttime; date = frstdate;
    output time_dat;
    output date1;
    event = 'Last  message ';
    time = lasttime; date = lastdate;
    output time_dat;
    output date2;
  end;

/*************************************************************************
* Construction of data set comprised of the relevant data                *
*************************************************************************/

/* Up to now, we have a data set containing the raw information, mainly
   the host-domain pair.
   (Remainder : what we call maybe improperly the host-domain pair is a
                string containing the host and then the list of possible
                subdomains, and then the domain, all of these separated
                by dots (i.e. host.subdom1.subdom2...subdomi.domain);
                the list of subdomains can be empty, and all is in
                upper case).
  We have to process these host-domain pairs, mainly do the neccessary
  remapping of host and domain names.  We will also spot error messages,
  or rather messages having an unknown address format or the ones having
  an unknown domain.  The illegal messages (messages using CERN gateways
  without having been allowed) will also be stored in a data set of their
  own */

data fine_dat (keep= hostdomo hostdomd nbytes obs_numb)
     illegal  (keep= date time nbytes type orig dest)
     err_data (keep= date time nbytes type orig dest);

    format country $20.;
    informat country $20.;

    set raw_data;
    /* since messages are normally logged twice when they cross the
       gateways (once when entering, once when leaving), so we choose to
       keep only one of these categories */

    if type~='export' and type~='deliver' then delete;

    obs_numb = 1;      /* this will be different for summary data */

    /* processing of the destination host-domain pair : we first have
       to decompose the hostdom variable in its components, namely the
       host (the name before the first dot), the domain (the name after
       the last dot), and possibly the list of subdomains (what is
       between the first and last dots).  When we have these components,
       we can do the remapping of host and domain names, and then we
       rebuild the host.subdom.domain names. */

    firstdot = index(hostdomd,'.');
    hd = hostdomd;
    do while (index(hd,'.') ~= 0);
       lastdot = index(hd,'.');
       if lastdot = 1
          then hd = substr(hd,2);
          else if lastdot = length(hd)
                  then hd = substr(hd,1,length(hd)-1);
                  else hd = substr(hd,1,lastdot-1) || '-' ||
                            substr(hd,lastdot+1);
    end;
    if firstdot > 1
       then host = substr(hostdomd,1,firstdot-1);
       else host = "";          /* for cases like user@.BITNET (real !) */
    if firstdot < lastdot-1     /* ... -1 for cases (real !) where we   */
                                /* have user@host..domain               */
       then between = substr(hostdomd,firstdot+1,lastdot-firstdot-1);
       else between = "";
    if lastdot ~= length(hostdomd)
       then domain = substr(hostdomd,lastdot+1);
       else domain = "";

    /* This will check if there is no remapping of host name */
    %include hostest;
    /* This will check if there is no remapping of the domain
       depending on host name */
    %include htdmtest;
    /* This will check if there is no remapping of the domain */
    %include domtest;

    /* this will give a value to country depending on the value of
       domain after remapping of the domain names.
       If domain is unknown, country will be set to 'UNKNOWN' */

    %include countmap;

    /* if the domain was unknown, then the country will have been
       unknown and the message will be output in the errors data set */

    if country='UNKNOWN' then do;
       output err_data;
       return;
    end;
    dom_dest = domain;

    /* now that we have remapped host and domain names, it's time
       to reassemble the host-domain entire name */

    if between = ""
       then hostdomd = trim(host)||'.'||trim(domain);
       else hostdomd = trim(host)||'.'||trim(between)||'.'||trim(domain);

    /* processing of the origin host-domain pair : for an explanation of
       this, please refer to the processing of the destination
       host-domain pair, here above */

    firstdot = index(hostdomo,'.');
    hd = hostdomo;
    do while (index(hd,'.') ~= 0);
       lastdot = index(hd,'.');
```

```
      if lastdot = 1
         then hd = substr(hd,2);
         else if lastdot = length(hd)
                 then hd = substr(hd,1,length(hd)-1);
                 else hd = substr(hd,1,lastdot-1) ||'-' ||
                           substr(hd,lastdot+1);
   end;
   if firstdot > 1 then host = substr(hostdomo,1,firstdot-1);
                   else host = "";
   if firstdot < lastdot -1
      then between = substr(hostdomo,firstdot+1,lastdot-firstdot-1);
      else between = "";
   if lastdot ~= length(hostdomo)
      then domain = substr(hostdomo,lastdot+1);
      else domain = "";

   %include hostest;
   %include htdmtest;
   %include domtest;
   %include countmap;

   if country='UNKNOWN' then do;
      output err_data;
      return;
   end;
   dom_orig = domain;
   if between = ""
      then hostdomo = trim(host)||'.'||trim(domain);
      else hostdomo = trim(host)||'.'||trim(between)||'.'||trim(domain);

   /* now that we have the values of the origin and destination domains
      and hosts, let's see if this message is legal, i.e. the sender or
      receiver domain is CERN, or the sender or receiver is explicitly
      allowed by CERN to use CERN's gateways */

   if dom_orig ~= 'CERN' and dom_dest ~= 'CERN' then do;
        %include chkallow;
        if ~allowed then output illegal;
   end;
   output fine_dat;


/*************************************************************************
* Adding the data just having been calculated to the data of the         *
* month and to the data of the year and summarizing them in the          *
* same data sets                                                         *
*************************************************************************/

/*************************************************************************
* Adding the main data to the monthly data                               *
*************************************************************************/

/* getting the old data with the brand new ones... */

data fine_all;
  set fine_dat monthsum.data;

/* ...and summarizing all this to get the new data set (updated version
   of the currently existing one) */

proc summary data=fine_all;
  class hostdomo hostdomd;
  var nbytes obs_numb;
  output out=finetmp sum=nbytes obs_numb;

/* dropping the unnecessary information brought in by the summary
   procedure and writing all this back on the old data set */

data monthsum.data (drop = _TYPE_ _FREQ_);
  set finetmp;
  if _TYPE_ = '11'B;

'*************************************************************************
  Adding the main data to the yearly data                               *
*************************************************************************/

data fine_all;
  set fine_dat yearsum.data;

proc summary data=fine_all;
  class hostdomo hostdomd;
  var nbytes obs_numb;
  output out=finetmp sum=nbytes obs_numb;

/* dropping the unnecessary information */

data yearsum.data (drop = _TYPE_ _FREQ_);
  set finetmp;
  if _TYPE_ = '11'B;

/*************************************************************************
* Adding the time to the monthly times                                   *
*************************************************************************/

data timetemp (drop = missing);
    set mnthtime.data time_dat;
    if time ~= missing;

proc sort data=timetemp;
    by date time;

data frsttime;
    set timetemp;
    if _N_ = 1;

proc sort data=timetemp;
    by descending date descending time;

data lasttime;
    set timetemp;
    if _N_ = 1;
```

```
data mnthtime.data;
    set lasttime frsttime;

proc sort data=mnthtime.data;
    by date time;

/*************************************************************************
* Adding the time to the yearly times                                    *
*************************************************************************/

data timetemp (drop = missing);
    set yeartime.data time_dat;
    if time ~= missing;

proc sort data=timetemp;
    by date time;

data frsttime;
    set timetemp;
    if _N_ = 1;

proc sort data=timetemp;
    by descending date descending time;

data lasttime;
    set timetemp;
    if _N_ = 1;

data yeartime.data;
    set lasttime frsttime;

proc sort data=yeartime.data;
    by date time;

/*************************************************************************
* Adding the types to the monthly types                                  *
*************************************************************************/

data weektype (keep = type type_nb);
    set raw_data;
    type_nb = 1;    /* number of observation of this type, will change */
                    /* when summarized with previous data              */

procedure summary data = weektype;
    class type;
    var type_nb;
    output out=typetmp1 sum=type_nb;

data type_all;
    set mnthtype.data weektype;

proc summary data = type_all;
    class type;
    var type_nb;
    output out=typetmp2 sum=type_nb;

data mnthtype.data (keep = type type_nb);
    set typetmp2;

/* number of messages exported during this period */
data exported (keep = exp_nb);
    set typetmp1;
    if type = 'export' then do;
      exp_nb = type_nb;
      output;
    end;

data expmsg (keep = frstdate lastdate exp_nb exp_avg);
    retain frstdate lastdate exp_nb;
    set date1;
    set date2;
    set exported;
    if frstdate ~= missing and lastdate ~= missing and exp_nb ~= missing
    then do;
        if frstdate ~= lastdate
            then exp_avg = exp_nb / (lastdate - frstdate);
            else exp_avg = exp_nb;
        output;
    end;

data exp_temp;
put 'exp_temp' _all_;
    set expmsg exp_msg.data;

data exp_msg.data;
put 'exp_msg.data' _all_;
    set exp_temp;

/*************************************************************************
* Adding the types to the yearly types                                   *
*************************************************************************/

data type_all;
    set yeartype.data weektype;

proc summary data = type_all;
    class type;
    var type_nb;
    output out=typetmp2 sum=type_nb;

data yeartype.data (keep = type type_nb);
    set typetmp2;

/*************************************************************************
* Adding the errors                                                      *
*************************************************************************/

data temp_err;
    set err_data montherr.data;

data montherr.data;
    set temp_err;
```

```
/*********************************************************************
* Adding the illegal messages                                        *
*********************************************************************/

data temp_ill;
    set illegal monthill.data;

data monthill.data;
    set temp_ill;


============================================================

/*********************************************************************
* SASMONTH : This is meant to be a SAS job to produce some statistics *
*            relative to EAN traffic from existing data sets on disk  *
*********************************************************************/

 options pagesize=110 linesize=120;
*options pagesize=40  linesize=79;

/* definition of the needed external files */

CMS FILEDEF COUNTMAP DISK DOMCOUNT TEST E; /* to get superdomain names */
CMS FILEDEF COSTMAP DISK COUNCOST TEST E;   /* countries' volume cost */
CMS FILEDEF MSGCOST DISK MSG_COST DATA E;   /* calculates cost of msg */
CMS FILEDEF CONFIG DISK CONFIG TEST E;      /* gives some config data */

/* Calculation of the cost of the messages */

data mnthsum2 (keep = hostdomo host_org dom_orig
                      hostdomd host_dst dom_dest
                      nbytes totcost obs_numb)
     mnthorg2 (keep = hostdomo obs_numb)
     mnthdst2 (keep = hostdomd obs_numb);

  format country $20.;
  informat country $20.;

  set monthsum.data;
  output mnthorg2;
  output mnthdst2;

  firstdot = index(hostdomo,'.');
  hd = hostdomo;
  do while (index(hd,'.') ~= 0);
     lastdot = index(hd,'.');
     hd = substr(hd,1,lastdot-1) || '-' || substr(hd,lastdot+1);
  end;
  host_org = substr(hostdomo,1,firstdot-1);
  dom_orig = substr(hostdomo,lastdot+1);

  firstdot = index(hostdomd,'.');
  hd = hostdomd;
  do while (index(hd,'.') ~= 0);
     lastdot = index(hd,'.');
     hd = substr(hd,1,lastdot-1) || '-' || substr(hd,lastdot+1);
  end;
  host_dst = substr(hostdomd,1,firstdot-1);
  dom_dest = substr(hostdomd,lastdot+1);

  /* calculation of the cost of the messages */

  domain = dom_dest;
  /* this will give a value to country depending on value of domain */
  %include countmap;
  /* this will give a value to Cvdom depending on value of country */
  %include costmap;
  /* this will give a value to totcost depending on value of Cvdom */
  %include msgcost;
  if act_cost='-' then totcost = -totcost;
  if domain = 'CERN' then totcost = 0;

  output mnthsum2;

/* Creation a synthesis of the above information by domains */

proc summary data = mnthsum2;
  class dom_orig dom_dest;
  var nbytes totcost obs_numb;
  output out=sum_data sum=nbytes totcost obs_numb;

/* Creation of 3 data sets containing respectively :
       - a summary of the origin domains
       - a summary of the destination domains
       - a summary of the domain to domain pairs
*/

data mnthorig (keep= obs_numb dom_orig)
     mnthdest (keep= obs_numb dom_dest)
     mnthdom  (keep= dom_orig dom_dest obs_numb nbytes bytemean totcost);
  set sum_data;
  bytemean = nbytes/obs_numb;

  if _TYPE_ = '01'B then output mnthdest;
  if _TYPE_ = '10'B then output mnthorig;
  if _TYPE_ = '11'B then output mnthdom;

/*********************************************************************
* Printing of the graph showing the progression in the daily         *
* average of exported messages                                       *
*********************************************************************/

proc plot data=exp_msg.data;
  plot exp_avg*lastdate='*' / vzero vpos=20;
  title Average number of messages exported per day;

/*********************************************************************
* Printing of table indicating first and last time of messages       *
*********************************************************************/
```

```
proc print data = mnthtime.data;
title Date and time of first and last records;


/*************************************************************************
* Type statistics                                                        *
*************************************************************************/

proc tabulate data=mnthtype.data format=12. order=freq;
  title Frequence of the types of records;
  class type;
  var type_nb;
  table type=' '*type_nb=' '*sum=' ' all='Total'*type_nb=' '*sum=' ';

/*************************************************************************
* Construction of data set summarizing the number of messages           *
* received, sent and exchanged by CERN hosts                             *
*************************************************************************/

data CERN_hst (keep = hostdom received sent xchanged);
  set mnthsum2;
  if dom_orig = 'CERN' then do;
       hostdom = hostdomo;
       sent = obs_numb; received = 0; xchanged = obs_numb;
       output;
   end;
  if dom_dest = 'CERN' then do;
       hostdom = hostdomd;
       received = obs_numb; sent = 0; xchanged = obs_numb;
       output;
   end;

proc summary data = CERN_hst;
  class hostdom;
  var received sent xchanged;
  output out=temp_hst sum=received sent xchanged;

proc sort data = temp_hst;
  by descending xchanged;

data tmp2_hst (keep = host received sent xchanged);
  set temp_hst;
  host = hostdom;

  /* Read the value of nb_cernh (the number of .CERN hosts to list)
     in the config file */
  %include config;

  if _N_ ~= 1 & _N_ <= nb_cernh+1;


proc print data = tmp2_hst;
  title Number of messages received, sent and exchanged by .CERN hosts;

/*************************************************************************
* Printing the most popular host-domain pairs as origin and then         *
* as destination                                                         *
*************************************************************************/

proc summary data=mnthorg2;
    class hostdomo;
    var obs_numb;
    output out = mnthorg3 sum = obs_numb;

proc sort data=mnthorg3;
    by descending obs_numb;

data temp_org (keep = orig obs_numb);
    set mnthorg3;
    orig = hostdomo;
    /* Read the value of nb_pop (the number of most popular domains to
       list in the config file */
    %include config;

    if _N_ ~= 1 & _N_ <= nb_pop+1;


proc print data=temp_org;
    title Most popular originating hosts;

proc summary data=mnthdst2;
    class hostdomd;
    var obs_numb;
    output out = mnthdst3 sum = obs_numb;

proc sort data=mnthdst3;
    by descending obs_numb;

data temp_dst (keep = dest obs_numb);
    set mnthdst3;
    dest = hostdomd;

    /* Read the value of nb_pop (the number of most popular domains to
       list in the config file */
    %include config;

    if _N_ ~= 1 & _N_ <= nb_pop+1;

proc print data=temp_dst;
    title Most popular destination hosts;

/*************************************************************************
* Printing a summary of the importance of the domains as origin and      *
* destinations                                                           *
*************************************************************************/

proc sort data=mnthorig;
    by descending obs_numb;

 proc print data=mnthorig;
 title Origin domains importance;
```

```
proc sort data=mnthdest;
    by descending obs_numb;

 proc print data=mnthdest;
 title Destination domains importance;

/*************************************************************************
* Printing of the observations related to the illegal messages          *
*************************************************************************/

data ill_temp (keep = miscinfo orig dest);
   format miscinfo $115.;
   informat miscinfo $115.;
   set monthill.data;
   d = put(date,date7.);
   t = put(time,time8.);
   n = put(nbytes,9.);
   miscinfo = d || "    " || t || "    " || type || n ||
          "                                                              .";

proc print data=ill_temp;
title Observations related to illegal messages;

/*************************************************************************
* Printing of the observations related to unknown address format or     *
* unknown domain name                                                   *
*************************************************************************/

data err_temp (keep = miscinfo orig dest);
   format miscinfo $115.;
   informat miscinfo $115.;
   set montherr.data;
   d = put(date,date7.);
   t = put(time,time8.);
   n = put(nbytes,9.);
   miscinfo = d || "    " || t || "    " || type || n ||
          "                                                              .";

proc print data=err_temp;
title Observations related to unknown addr format or unknown domain name;

  /*************************************************************************
  Construction of summary data between superdomains                      *
*************************************************************************/

data monthsup (drop = host_org dom_orig host_dst dom_dest);
  format sup_orig sup_dest superdom $20.;
  set mnthsum2;
  domain = dom_orig;
  %include countmap;
  sup_orig = superdom;
  domain = dom_dest;
  %include countmap;
  sup_dest = superdom;

proc summary data = monthsup;
  class sup_orig sup_dest;
  var nbytes totcost obs_numb;
  output out=temp_sup sum=nbytes totcost obs_numb;

data sup_data (keep = sup_orig sup_dest
                      nbytes bytemean totcost obs_numb);
  set temp_sup;
  bytemean = nbytes/obs_numb;
  if _TYPE_ = '11'B;

/*************************************************************************
* Traffic between superdomains                                          *
* (Number of messages only)                                             *
*************************************************************************/

proc tabulate data=sup_data  format=9. noseps;
  title Summary of traffic between Superdomains (Summary);
  class sup_orig sup_dest;
  var obs_numb nbytes bytemean totcost;
  table sup_orig='Source Superdomains  ----------'
           all='------------ Total',
        sup_dest='Destination Superdomains'*obs_numb=' '*sum=' '
        all='Total'*obs_numb=' '*sum=' '
        / rts = 15 condense;

/*************************************************************************
* Traffic between superdomains                                          *
* (extended form)                                                       *
*************************************************************************/

proc tabulate data=sup_data  format=9. noseps;
  title Summary of traffic between Superdomains (extended);
  class sup_orig sup_dest;
  var obs_numb nbytes bytemean totcost;
  table sup_orig='Source Superdomains --------' all='------------ Total',
        sup_dest='Destination Superdomains'*
                        (obs_numb=' '*(sum='N Calls' pctsum='Percent')
                         nbytes=' '*sum='Bytes'
                         bytemean=' '*sum='B Mean'
                         totcost='Cost SF'*sum=' ')
        all='Total'*
                        (obs_numb=' '*(sum='N Calls' pctsum='Percent')
                         nbytes=' '*sum='Bytes')
        / rts = 15 condense;

/*************************************************************************
* Traffic between all domains sorted by alpha order                     *
* (extended form)                                                       *
*************************************************************************/

proc tabulate data=mnthdom  format=9. noseps;
  title Summary of traffic between all domains (extended form);
  class dom_orig dom_dest;
  var obs_numb nbytes bytemean totcost;
  table dom_orig='Source Domain     -------' all='------------ Total',
        dom_dest='Destination Domain'*
```

```
                        (obs_numb=' '*(sum='N Calls' pctsum='Percent')
                         nbytes=' '*sum='Bytes'
                         bytemean=' '*sum='B Mean'
                         totcost='Cost SF'*sum=' ')
        all='Total'*
                        (obs_numb=' '*(sum='N Calls' pctsum='Percent')
                         nbytes=' '*sum='Bytes')
        / rts = 13 condense;


============================================================

/*********************************************************************
* SASYEAR  : This is meant to be a SAS job to produce some statistics *
*            relative to EAN traffic from the yearly EAN data sets    *
*********************************************************************/

 options pagesize=110 linesize=120;
*options pagesize=40  linesize=79;

/* definition of the needed external files */

CMS FILEDEF COUNTMAP DISK DOMCOUNT TEST E; /* to get superdomain names */
CMS FILEDEF COSTMAP DISK COUNCOST TEST E;   /* countries' volume cost */
CMS FILEDEF MSGCOST DISK MSG_COST DATA E;   /* calculates cost of msg */
CMS FILEDEF CONFIG DISK CONFIG TEST E;      /* gives some config data */

/* Calculation of the cost of the messages */

data yearsum2 (keep = hostdomo host_org dom_orig
                      hostdomd host_dst dom_dest
                      nbytes totcost obs_numb)
     yearorg2 (keep = hostdomo obs_numb)
     yeardst2 (keep = hostdomd obs_numb);

  format country $20.;
  informat country $20.;

  set yearsum.data;
  output yearorg2;
  output yeardst2;

  firstdot = index(hostdomo,'.');
  hd = hostdomo;
  do while (index(hd,'.') ^= 0);
     lastdot = index(hd,'.');
     hd = substr(hd,1,lastdot-1) || '-' || substr(hd,lastdot+1);
  end;
  host_org = substr(hostdomo,1,firstdot-1);
  dom_orig = substr(hostdomo,lastdot+1);

  firstdot = index(hostdomd,'.');
  hd = hostdomd;
  do while (index(hd,'.') ^= 0);
     lastdot = index(hd,'.');
     hd = substr(hd,1,lastdot-1) || '-' || substr(hd,lastdot+1);
  end;
  host_dst = substr(hostdomd,1,firstdot-1);
  dom_dest = substr(hostdomd,lastdot+1);

  /* calculation of the cost of the messages */

  domain = dom_dest;
  /* this will give a value to country depending on value of domain */
  %include countmap;
  /* this will give a value to Cvdom depending on value of country */
  %include costmap;
  /* this will give a value to totcost depending on value of Cvdom */
  %include msgcost;
  if act_cost='-' then totcost = -totcost;
  if domain = 'CERN' then totcost = 0;

  output yearsum2;

/* Creation a synthesis of the above information by domains */

proc summary data = yearsum2;
  class dom_orig dom_dest;
  var nbytes totcost obs_numb;
  output out=sum_data sum=nbytes totcost obs_numb;

/* Creation of 3 data sets containing respectively :
      - a summary of the origin domains
      - a summary of the destination domains
      - a summary of the domain to domain pairs
*/

data yearorig (keep= obs_numb dom_orig)
     yeardest (keep= obs_numb dom_dest)
     yeardom  (keep= dom_orig dom_dest obs_numb nbytes bytemean totcost);
  set sum_data;
  bytemean = nbytes/obs_numb;

  if _TYPE_ = '01'B then output yeardest;
  if _TYPE_ = '10'B then output yearorig;
  if _TYPE_ = '11'B then output yeardom;

/*********************************************************************
* Printing of the graph showing the progression in the daily         *
* average of exported messages                                       *
*********************************************************************/

proc plot data=exp_msg.data;
  plot exp_avg*lastdate='*' / vzero vpos=20;
  title Average number of messages exported per day;

/*********************************************************************
* Printing of table indicating first and last time of messages       *
*********************************************************************/

proc print data = yeartime.data;
title Date and time of first and last records;
```

```
/*************************************************************************
* Type statistics                                                        *
*************************************************************************/

proc tabulate data=yeartype.data format=12. order=freq;
  title Frequence of the types of records;
  class type;
  var type_nb;
  table type=' '*type_nb=' '*sum=' ' all='Total'*type_nb=' '*sum=' ';

/*************************************************************************
* Construction of data set summarizing the number of messages           *
* received, sent and exchanged by CERN hosts                             *
*************************************************************************/

data CERN_hst (keep = hostdom received sent xchanged);
  set yearsum2;
  if dom_orig = 'CERN' then do;
       hostdom = hostdomo;
       sent = obs_numb; received = 0; xchanged = obs_numb;
       output;
   end;
  if dom_dest = 'CERN' then do;
       hostdom = hostdomd;
       received = obs_numb; sent = 0; xchanged = obs_numb;
       output;
   end;

proc summary data = CERN_hst;
  class hostdom;
  var received sent xchanged;
  output out=temp_hst sum=received sent xchanged;

proc sort data = temp_hst;
  by descending xchanged;

data tmp2_hst (keep = host received sent xchanged);
  set temp_hst;
  host = hostdom;

  /* Read the value of nb_cernh (the number of .CERN hosts to list)
     in the config file */
  %include config;

  if _N_ ~= 1 & _N_ <= nb_cernh+1;


proc print data = tmp2_hst;
  title Number of messages received, sent and exchanged by .CERN hosts;

/*************************************************************************
* Printing the most popular host-domain pairs as origin and then         *
* as destination                                                         *
*************************************************************************/

proc summary data=yearorg2;
    class hostdomo;
    var obs_numb;
    output out = yearorg3 sum = obs_numb;

proc sort data=yearorg3;
    by descending obs_numb;

data temp_org (keep = orig obs_numb);
    set yearorg3;
    orig = hostdomo;
    /* Read the value of nb_pop (the number of most popular domains to
       list in the config file */
    %include config;

    if _N_ ~= 1 & _N_ <= nb_pop+1;


proc print data=temp_org;
    title Most popular originating hosts;

proc summary data=yeardst2;
    class hostdomd;
    var obs_numb;
    output out = yeardst3 sum = obs_numb;

proc sort data=yeardst3;
    by descending obs_numb;

data temp_dst (keep = dest obs_numb);
    set yeardst3;
    dest = hostdomd;

    /* Read the value of nb_pop (the number of most popular domains to
       list in the config file */
    %include config;

    if _N_ ~= 1 & _N_ <= nb_pop+1;

proc print data=temp_dst;
    title Most popular destination hosts;

/*************************************************************************
* Printing a summary of the importance of the domains as origin and      *
* destinations                                                           *
*************************************************************************/

proc sort data=yearorig;
    by descending obs_numb;

 proc print data=yearorig;
 title Origin domains importance;

proc sort data=yeardest;
    by descending obs_numb;
```

```
 proc print data=yeardest;
 title Destination domains importance;

/*************************************************************************
* Construction of summary data between superdomains                      *
*************************************************************************/

data yearhsup (drop = host_org dom_orig host_dst dom_dest);
  format sup_orig sup_dest superdom $20.;
  set yearsum2;
  domain = dom_orig;
  %include countmap;
  sup_orig = superdom;
  domain = dom_dest;
  %include countmap;
  sup_dest = superdom;

proc summary data = yearhsup;
  class sup_orig sup_dest;
  var nbytes totcost obs_numb;
  output out=temp_sup sum=nbytes totcost obs_numb;

data sup_data (keep = sup_orig sup_dest
                      nbytes bytemean totcost obs_numb);
  set temp_sup;
  bytemean = nbytes/obs_numb;
  if _TYPE_ = '11'B;

/*************************************************************************
* Traffic between superdomains                                           *
* (Number of messages only)                                              *
*************************************************************************/

proc tabulate data=sup_data  format=9. noseps;
  title Summary of traffic between Superdomains (Summary);
  class sup_orig sup_dest;
  var obs_numb nbytes bytemean totcost;
  table sup_orig='Source Superdomains  ----------'
            all='------------ Total',
        sup_dest='Destination Superdomains'*obs_numb=' '*sum=' '
        all='Total'*obs_numb=' '*sum=' '
        / rts = 15 condense;

/*************************************************************************
* Traffic between superdomains                                           *
* (extended form)                                                        *
*************************************************************************/

proc tabulate data=sup_data  format=9. noseps;
  title Summary of traffic between Superdomains (extended);
  class sup_orig sup_dest;
  var obs_numb nbytes bytemean totcost;
  table sup_orig='Source Superdomains --------' all='------------ Total',
        sup_dest='Destination Superdomains'*
                        (obs_numb=' '*(sum='N Calls' pctsum='Percent')
                         nbytes=' '*sum='Bytes'
                         bytemean=' '*sum='B Mean'
                         totcost='Cost SF'*sum=' ')
        all='Total'*
                        (obs_numb=' '*(sum='N Calls' pctsum='Percent')
                         nbytes=' '*sum='Bytes')
        / rts = 15 condense;

/*************************************************************************
* Traffic between all domains sorted by alpha order                      *
* (extended form)                                                        *
*************************************************************************/

proc tabulate data=yeardom  format=9. noseps;
  title Summary of traffic between all domains (extended form);
  class dom_orig dom_dest;
  var obs_numb nbytes bytemean totcost;
  table dom_orig='Source Domain     -------' all='------------ Total',
        dom_dest='Destination Domain'*
                        (obs_numb=' '*(sum='N Calls' pctsum='Percent')
                         nbytes=' '*sum='Bytes'
                         bytemean=' '*sum='B Mean'
                         totcost='Cost SF'*sum=' ')
        all='Total'*
                        (obs_numb=' '*(sum='N Calls' pctsum='Percent')
                         nbytes=' '*sum='Bytes')
        / rts = 13 condense;


============================================================

/*************************************************************************
* YEARINIT : This SAS program initializes the data accumulated in the *
*            yearly data sets related to the EAN statistic analysis   *
*************************************************************************/

CMS FILEDEF VOIDFILE DISK DOESNT EXIST A;
options replace;

/*************************************************************************
* Initialisation of data set who will contain the summary data for       *
* the current year                                                       *
*************************************************************************/

data yearsum.data (keep= hostdomo hostdomd nbytes obs_numb);
    infile voidfile;
    format hostdomo hostdomd $70.;
    delete;

/*************************************************************************
* Initialisation of data set who will contain the time of the first      *
* and last observation of the current year                               *
*************************************************************************/

data yeartime.data (keep = event time date);
  infile cards missover;
```

```
  informat event $15.;
  informat date DATE7.;
  informat time time8.;
  format event $15.;
  format date DATE7.;
  format time time8.;
  input event time date;
  cards;
First_time
Last_time
;

/*************************************************************************
* Initialisation of data set who will contain the type frequence of     *
* the records of the year                                               *
*************************************************************************/

data yeartype.data (keep= type type_nb);
    infile voidfile;
    delete;

/*************************************************************************
* Initialisation of data set who will contain the progression of the    *
* average number of messages sent per day                               *
*************************************************************************/

data exp_msg.data (keep= frstdate lastdate exp_nb exp_avg);
    infile voidfile;
    delete;


============================================================

/*************************************************************************
* MNTHINIT : This SAS program initializes the monthly data contained    *
*            in the data sets related to the EAN statistic application*
*************************************************************************/

CMS FILEDEF VOIDFILE DISK DOESNT EXIST A;
options replace;

  /*************************************************************************
  Initialisation of data set who will contain the summary data for      *
* the current month                                                     *
*************************************************************************/

data monthsum.data (keep= hostdomo hostdomd nbytes obs_numb);
    infile voidfile;
    format hostdomo hostdomd $70.;
    delete;      /* necessary to ensure there is no observation */

/*************************************************************************
* Initialisation of data set who will contain the time of the first     *
* and last observation of the current month                             *
*************************************************************************/

data mnthtime.data (keep = event time date);
  infile cards missover;
  informat event $15.;
  informat date DATE7.;
  informat time time8.;
  format event $15.;
  format date DATE7.;
  format time time8.;
  input event time date;
  cards;
First_time
Last_time
;

/*************************************************************************
* Initialisation of data set who will contain the type frequence of     *
* the records of the month                                              *
   *************************************************************************/

data mnthtype.data (keep= type type_nb);
    infile voidfile;
    delete;

/*************************************************************************
* Initialisation of data set who will contain the problematic records   *
* of the month : unknown addressing format                              *
*************************************************************************/

data montherr.data (keep = date time nbytes type orig dest);
    infile voidfile;
    format date DATE7.;
    format time time8.;
    format orig dest type $100.;
    delete;

/*************************************************************************
* Initialisation of data set who will contain the problematic records   *
* of the month : illegal use of CERN's gateways                         *
*************************************************************************/

data monthill.data (keep = date time nbytes type orig dest);
    infile voidfile;
    format date DATE7.;
    format time time8.;
    format orig dest type $100.;
    delete;


============================================================

/* SASBATCH : This REXX procedure is meant to control the batch job
              whose goal is to :
              - each nb_days_to_wait'th day, get EAN files produced on a
                VAX, analyse these files, and produce SAS data sets which
                will keep a summary of the information contained in the
                raw files and add these SAS data to the permanent data
```

```
                sets accumulating the summary information for the current
                month and year.
              - each nb_times_to_wait'th time it is executed,
                produce a report from the data accumulated
                in the monthly permanent data set and send that report
                to the person who will have a look at it...
*/

responsible = "MAIL-SUPPORT VXGIFT.CERN"

nb_days_to_wait = 7   /* number of days to wait before analysing and */
                      /* and producing a summary of the accumulated  */
                      /* data, but no report                         */

nb_times_to_wait = 1  /* number of times to analyse the accumulated  */
                      /* and produce summaries before setting up a   */
                      /* report summarizing all that information     */


DISK_LOCKED = 3                 /* error 3 : disk was write locked  */

trace r               /* trace it so you can check the execution in  */
                      /* the batch logfile */

address command

/* Access the minidisks needed */

"EXEC GIME EMSTAT 192 D dummy (MR"

/* If you can't get access to the disk containing the data (surely
   because it's write locked), issue a warning to the responsible
   and simply resubmit the job for the next time
*/

if rc ~= 0 then call problem DISK_LOCKED

/* Process the files by the SAS package and add the resulting information
   to the data sets already built (the SAS job file is first copied on
   minidisk A (of the batch machine) in order for the resulting files to
   be also output on the A disk) */

"COPY SASWEEK SAS * SASWEEKB SAS A"
"EXEC SASWEEKB"

/* if something went wrong, exit with to code returned from SASWEEKB */

if rc ~= 0 then call problem rc

/* the input file has been consumed, so delete it (or rather rename it to
   keep it until the next time a file is processed, just in case...)
*/

"ERASE EANFILE OLDDATA D"
"RENAME EANFILE DATA D EANFILE OLDDATA D"


/* if this is the right time, produce monthly reports... */

/* if the file containing the number of times the batch job has been
   running since the last initialization of the value contained in that
   file does not exist, then the batch job was never run before (or the
   situation is comparable).  In that case, just create the file with an
   initial value of 0
*/

if ~fexist("TIMES DATA D") then do
   queue "COMMAND I 0"
   queue "COMMAND FILE"
   "XEDIT TIMES DATA D"
end

   read the value in the file */

"EXECIO 1 DISKR TIMES DATA D (VAR TIMES"
times = times + 1

/* time to produce the report ? */

if times >= nb_times_to_wait then do

/* YES, so let's go and produce it... */

     "COPY SASMONTH SAS * SASMNTHB SAS A (REP"
     "EXEC SAS SASMNTHB"

/* print the file containing the results of the analysis */

     "EXEC PR3812 SASMNTHB LISTING (PRTR=DD31A FORMS=P1C"

/* Create the file containing a summary of the report */

     "COPY SASMAIL SAS * SASMAILB SAS A (REP"
     "EXEC SAS SASMAILB"

/* put the resulting file in a format more suitable for screen reading */

     "XEDIT SASMAILB LISTING (PROFILE SASMAIL NOMSG"

/* ... and send it to the person who will analyse these data */

     "EXEC SENDGATE SASMAILB LISTING A" responsible

/* Reset the monthly data, i.e. initialise the monthly SAS data sets and
   reset the value in the file TIMES DATA D to 0 */

     "COPY MNTHINIT SAS * MNTINITB SAS A (REP"
     "EXEC SAS MNTINITB"
     times = 0
end
```

```
/* write the value of times back to its file */

"EXECIO 1 DISKW TIMES DATA D 1 (VAR TIMES"

/* Resubmit the job for next time. The options are :
      - CLASS S              : maximum duration = 4 min CPU
      - CPU CERNVM           : use of the CERNVM CPU
      - RUNDATE...           : run nb_days_to_wait from today
      - RUNTIME '12:30'      : run at 12.30 am
      - NORETURN...          : don't return the following files
*/

"EXEC BATCH SUBMIT
     (CLASS S
      CPU CERNVM
      RUNDATE '" substr(xdate(I,+nb_days_to_wait),1,6) "'
      RUNTIME '12:30'
      NORETURN '* DATA'   NORETURN '* SAS'       NORETURN '* NETLOG'
      NORETURN '* SASLOG' NORETURN '* SASMAILB'
    ) CALLBTCH"

/* TEST PURPOSE ONLY : */
/*
t = time(m)
t = t + 1
hh = (t % 60) // 24
mm = t // 60
if mm < 10 then mm = '0' || value(mm)

"EXEC BATCH SUBMIT (CLASS S CPU CERNVM RUNTIME" hh || ":" || mm "
    NORETURN '* DATA'   NORETURN '* SAS'       NORETURN '* NETLOG'
    NORETURN '* SASLOG' NORETURN 'SASMAILB *'
    ) CALLBTCH"
exit
*/

/* when the batch job could not get all the resources it needs to
   complete its task, it sends an error message to the responsible
   and resubmits itself for the day after */

problem:

   arg err_code

   "EXEC WARNING" err_code responsible
   "EXEC BATCH SUBMIT
       (CLASS S
        CPU CERNVM
        RUNDATE '" substr(xdate(I,+1),1,6) "'
        RUNTIME '12:30'
        NORETURN '* DATA'   NORETURN '* SAS'       NORETURN '* NETLOG'
        NORETURN '* SASLOG' NORETURN 'SASMAILB *'
      ) CALLBTCH"
   exit

============================================================

/* CALLBTCH EXEC : This exec has only one goal : to call the SASBATCH exec !
                   This way of calling SASBATCH, which is supposed to run
                   in batch job, allows the modifications to SASBATCH to be
                   taken into account the very next time the batch job will
                   be run
*/

"EXEC SASBATCH"

============================================================

/* SASWEEKB EXEC :
   This exec controls the analysis of the EAN log file (normally
   EANFILE DATA D).  For this, it has first to check that all the
   files necessary to the proper execution of the SAS program are
   present


trace c                          /* to be able to understand what went */
                                 /* wrong during the batch job (if ever*/
                                 /* that happens, of course !!)        */
address command

tempdisk = "E"                   /* temporary work disk */
cyl_nb = 50                      /* number of cylinders asked for the  */
                                 /* temporary work disk */

retries = 5                      /* number of retries to do if I can't */
                                 /* get work space at the first time   */
mm =  20                         /* number of minutes to wait before   */
                                 /* trying to get the work space again */

ERR_SPACE = 1                    /* error  : not enough work space     */
WNG_SPACE = 2                    /* warning : not enough work space    */
                                 /*           I retry in mm minutes    */
NO_INPUT_FILE = 4                /* error : input file not found       */

/* Access the SAS minidisk (the EMSTAT data disk should have been
   accessed by the exec calling this one (SASBATCH normally))
*/

if ~fexist("SAS EXEC") then "EXEC GIME SAS"
emdisk = substr(qfile("EMSTAT DISK","FMODE"),1,1)

/* get some space for SAS to process the data */
i = 1
do while (~qdisk(tempdisk,accessed))
    "EXEC GIME" cyl_nb tempdisk
    if rc ~= 0 then do
       if i <= retries then do
          "EXEC WARNING" WNG_SPACE
          "CP SLEEP" mm "MIN"
          i = i + 1
       end
       else exit ERR_SPACE
```

```
    end
end

if ~fexist("EANFILE DATA D") then exit NO_INPUT_FILE

/* Creation of the SAS statements allowing a remapping of the domains
   and included in the SAS job */

  "COPY HOST DATA D HOST TEST" tempdisk "(REP"
  queue "HOST"
  "XEDIT HOST TEST" tempdisk

  "COPY DOMAIN DATA D DOMAIN TEST" tempdisk "(REP"
  queue "DOMAIN"
  "XEDIT DOMAIN TEST" tempdisk

  "COPY NOTCERN DATA D NOTCERN TEST" tempdisk "(REP"
  queue "NOTCERN"
  "XEDIT NOTCERN TEST" tempdisk

  "COPY HOSTDOM DATA D HOSTDOM TEST" tempdisk "(REP"
  queue "HOSTDOM"
  "XEDIT HOSTDOM TEST" tempdisk

  "COPY DOMCOUNT DATA D DOMCOUNT TEST" tempdisk "(REP"
  queue "DOMCOUNT"
  "XEDIT DOMCOUNT TEST" tempdisk

  "COPY COUNCOST DATA D COUNCOST TEST" tempdisk "(REP"
  queue "COUNCOST"
  "XEDIT COUNCOST TEST" tempdisk

  "COPY ALLOWED DATA D ALLOWED TEST" tempdisk "(REP"
  queue "ALLOWED"
  "XEDIT ALLOWED TEST" tempdisk

  "COPY CONFIG DATA D CONFIG TEST" tempdisk "(REP"
  queue "CONFIG"
  "XEDIT CONFIG TEST" tempdisk

  "COPY MSG_COST DATA D = =" tempdisk "(REP"

  ace c

/* Process the files by the SAS package and add the resulting information
   to the data sets already built */

"EXEC SAS SASWEEKB"

============================================================

/* WARNING EXEC : this exec must warn the responsible when something went
                  wrong with the execution of the batch job analysing the
                  EAN log files */


arg err_code responsible

ERR_SPACE = 1                    /* error  : not enough work space     */
WNG_SPACE = 2                    /* warning : not enough work space    */
                                 /*           I retry in mm minutes    */
DISK_LOCKED = 3                  /* data disk is locked                */
NO_INPUT_FILE = 4                /* error : no input file              */


fn = "ERROR"; ft = "TEMP$$"; fm = "A"

address command

msg.1 = "It's" time() "on" date("E")
msg.2 = "There was a problem with the SAS BATCH JOB :"

 'lect
    when err_code = ERR_SPACE then call error_space
    when err_code = WNG_SPACE then call wng_space
    when err_code = DISK_LOCKED then call err_locked
    when err_code = NO_INPUT_FILE then call err_no_file
    otherwise call unknown_error
end

queue "COMMAND SET CASE M I"
do i = 1 to msg.0
   queue "COMMAND I" msg.i
end
queue "COMMAND FILE"
"XEDIT" fn ft fm
"EXEC SENDGATE" fn ft fm responsible
"ERASE" fn ft fm

exit 0

error_space:
    msg.3 = "I could not get enough space to work !"
    msg.4 = "(Temporary minidisk)"
    msg.5 = "Sorry, I abort... but just for the moment !!"
    msg.6 = "I will retry tomorrow..."
    msg.0 = 6
    return

wng_space:
    msg.3 = "I could not get enough space to work !"
    msg.4 = "(Temporary minidisk)"
    msg.5 = "I retry in a few minutes..."
    msg.0 = 5
    return

err_locked:
    msg.3 = "I could not access the data disk (normally disk D of"
    msg.4 = "EMSTAT) because, I suppose, it was WRITE-LOCKED by someone"
    msg.5 = "else.  I did no data processing this time and only"
    msg.6 = "resubmitted the batch job for tomorrow (hoping by that"
    msg.7 = "time, the disk will have been released...)"
```

```
    msg.0 = 7
    return

err_no_file:
    msg.3 = "I could not find the input file (norrmally EANFILE DATA D"
    msg.4 = "on EMSTAT) because, I suppose, there was a problem during"
    msg.5 = "the transfer (INTERLINK problem)."
    msg.6 = "I did no data processing this time and only"
    msg.7 = "resubmitted the batch job for tomorrow (hoping by that"
    msg.8 = "time, I will find an input data file...)"
    msg.0 = 8
    return

unknown_error:
    msg.3 = "Unknown error !!"
    msg.4 = "(But certainly a programmer error...Really don't understand"
    msg.5 = " what happened ???)"
    msg.0 = 5
    return


============================================================

/* INITALL : This exec initializes all the data contained in the
             SAS data sets related to the EAN statistics application */

address command

say "All the EAN data (monthly and yearly) will be erased !!!"
say "Continue ? (Y/N)"
pull answer
answer = translate(answer)
if answer ¬= 'Y' then do
   say "Ok, nothing done."
   exit
end
say "Ok, initializing..."
"EXEC GIME EMSTAT 192 D (MR QUIET"
if ¬fexist("SAS EXEC") then "EXEC GIME SAS (QUIET"
"EXEC SAS MNTHINIT"
"EXEC SAS YEARINIT"
  ?ASE TIMES DATA D"
 -XEC DROP D (QUIET"
say "All the EAN data sets have been initialized..."
exit


============================================================

/* MNTHINIT : This exec initializes the monthly data (current data)
   contained in the SAS data sets related to the EAN statistics
   application */

address command

say "The EAN monthly data will be erased !!!"
say "Continue ? (Y/N)"
pull answer
answer = translate(answer)
if answer ¬= 'Y' then do
   say "Ok, nothing's done."
   exit
end
say "Ok, initializing..."
"EXEC GIME EMSTAT 192 D (MR QUIET"
if ¬fexist("SAS EXEC") then "EXEC GIME SAS (QUIET"
"EXEC SAS MNTHINIT"
"EXEC DROP D (QUIET"
say "The EAN monthly data sets have been initialized..."
exit


 ============================================================

/* YEARINIT : This exec initializes the yearly data contained in the
              SAS data sets related to the EAN statistics application */

address command

say "The EAN yearly data will be erased !!!"
say "Continue ? (Y/N)"
pull answer
answer = translate(answer)
if answer ¬= 'Y' then do
   say "Ok, nothing's done."
   exit
end
say "Ok, initializing..."
"EXEC GIME EMSTAT 192 D (MR QUIET"
if ¬fexist("SAS EXEC") then "EXEC GIME SAS (QUIET"
"EXEC SAS YEARINIT"
"EXEC DROP D (QUIET"
say "The EAN yearly data sets have been initialized..."
say "Happy new year !"
exit


============================================================

/* SBATCH EXEC :
   This exec submits the EAN batch job.  Arguments are the date and the
   time the batch is to be submitted.  If no argument is given, it's
   submitted right now.  If the date only is given (format MM/DD), then
   it's submitted at 7.30 am for MM/DD.  If the date and the time (format
   hh:mm) are both specified, then the batch job is submitted at that
   time at that day.
*/

arg date time
if date = "" then rundate = ""
             else rundate = "RUNDATE" date
if date = "" then runtime = ""
```

```
            else if time = "" then runtime = "RUNTIME '07:30'"
                              else runtime = "RUNTIME '"time"'"

"EXEC DROP D (QUIET"  /* to be sure minidisk D won't be write locked
                         at the critical moment... */
"EXEC BATCH SUBMIT
    (CPU CERNVM
     CLASS S
     NORETURN '* SAS'   NORETURN '* SASLOG'   NORETURN '* NETLOG'
     NORETURN '* DATA'  NORETURN 'SASMAILB *'
     " rundate runtime "
     ) CALLBTCH"

============================================================

/* HOST Xedit : builds a file of SAS conditionnal SAS statement from
   a text data file containing the mapping of hosts into hosts */
"COMMAND EXTRACT /SIZE"
filesize = size.1

"COMMAND TOP"
"COMMAND DOWN"
do filesize
  "COMMAND STACK"
  "COMMAND DEL"
  parse upper pull host1 host2 .
  if substr(host1,1,1) ¬= '*' then do
     "COMMAND UP"
     "COMMAND I if host = '" || host1 || "'"
     "COMMAND I then host = '" || host2 || "'; else"
     "COMMAND DOWN"
  end
end
"COMMAND I ;"
"COMMAND FILE"

============================================================

/* DOMAIN Xedit : builds a file of SAS conditionnal SAS statement from
   a text data file */
"COMMAND EXTRACT /SIZE"
  lesize = size.1

"COMMAND TOP"
"COMMAND DOWN"
do filesize
  "COMMAND STACK"
  "COMMAND DEL"
  parse upper pull domain1 domain2 .
  if substr(domain1,1,1) ¬= '*' then do
     "COMMAND UP"
     "COMMAND I if domain = '" || domain1 || "'"
     "COMMAND I then domain = '" || domain2 || "'; else"
     "COMMAND DOWN"
  end
end
"COMMAND I ;"
"COMMAND FILE"

============================================================

/* HOSTDOM Xedit : builds a file of SAS conditionnal SAS statement from
   a text data file containing the mapping of domains into domains
   depending of the value of the corresponding host */

"COMMAND EXTRACT /SIZE"
filesize = size.1

"COMMAND TOP"
"COMMAND DOWN"
do filesize
  "COMMAND STACK"
  "COMMAND DEL"
  parse upper pull maphost mapdom .
  if substr(maphost,1,1) ¬= '*' then do
     "COMMAND UP"
     "COMMAND I if host = '" || maphost || "'"
     "COMMAND I then domain = '" || mapdom || "'; else"
     "COMMAND DOWN"
  end
end
"COMMAND I ;"
"COMMAND FILE"

============================================================

/* YEAR Xedit : builds the SAS statement giving it's value to the
   variable year from the file year data */

"COMMAND EXTRACT /SIZE"
filesize = size.1

value_nb = 1                        /* number of the next value to read */

"COMMAND TOP"
"COMMAND DOWN"
do filesize
  "COMMAND STACK"
  "COMMAND DEL"
  parse upper pull value .
  if substr(value,1,1) ¬= '*' then do
     "COMMAND UP"
     if value_nb = 1 then "COMMAND I year = " value ";"
     if value_nb = 2 then "COMMAND I nb_pop = " value ";"
     if value_nb = 3 then "COMMAND I nb_cernh = " value ";"
     value_nb = value_nb + 1
     "COMMAND DOWN"
  end
end
"COMMAND FILE"
```

```
============================================================

/* NOTCERN Xedit : builds a file of SAS conditionnal statements from
   a text file containing the names of domains from which the .CERN
   suffix should be removed */

"COMMAND EXTRACT /SIZE"
filesize = size.1

"COMMAND TOP"
"COMMAND DOWN"
do filesize
  "COMMAND STACK"
  "COMMAND DEL"
  parse upper pull notcerndom .
  if substr(notcerndom,1,1) ¬= '*' then do
   "COMMAND UP"
   "COMMAND I if index(hostdom,'."notcerndom".CERN>') ¬= 0"
   "COMMAND I then hostdom = substr(hostdom,1,index(hostdom,'.CERN>')-1)"
   "COMMAND I                ||'>'; else"
   "COMMAND DOWN"
  end
end
"COMMAND I ;"
"COMMAND FILE"

============================================================

/* ALLOWED Xedit : builds a file of SAS conditionnal statements from
   a text file containing the names of domains allowed to use the CERN
   gateways. */

"COMMAND EXTRACT /SIZE"
filesize = size.1

"COMMAND TOP"
"COMMAND I allowed = 0;"
"COMMAND DOWN"
do filesize
  "COMMAND STACK"
  "COMMAND DEL"
  parse upper pull domain .
  if substr(domain,1,1) ¬= '*' then do
     "COMMAND UP"
     "COMMAND I if dom_orig = '"||domain||"' or dom_dest ='"||domain||"'"
     "COMMAND I   then allowed = 1; else"
     "COMMAND DOWN"
  end
end
"COMMAND I ;"
"COMMAND FILE"

============================================================

/* YEAR Xedit : builds the SAS statement giving it's value to the
   variable year from the file year data */

"COMMAND EXTRACT /SIZE"
filesize = size.1

value_nb = 1                        /* number of the next value to read */

"COMMAND TOP"
"COMMAND DOWN"
do filesize
  "COMMAND STACK"
  "COMMAND DEL"
  parse upper pull value .
  if substr(value,1,1) ¬= '*' then do
     "COMMAND UP"
     if value_nb = 1 then "COMMAND I year = " value ";"
     if value_nb = 2 then "COMMAND I nb_pop = " value ";"
     if value_nb = 3 then "COMMAND I nb_cernh = " value ";"
     value_nb = value_nb + 1
     "COMMAND DOWN"
  end
end
"COMMAND FILE"


============================================================

* HOST DATA :
* This file contains the mapping of several hosts, in connection with
* the SAS job producing statistics about the EAN traffic
* To add another host mapping, simply edit the file and add the
* relevant line, which will be composed of 2 words (no imbedded blank!) :
* the first one is the name of the host appearing in the log files
* the second one being the name that you want to appear in the report
*  NOTES : - as you can imagine, each line begining with '*' is a comment
*          - the host names will be UPPER CASEd anyway
*          - this file will be processed by the Xedit macro "host xedit"
*            to produce a file of SAS statements of the following form :
*                if host = 'orig_host1' then host = 'map_host1'; else
*                if host = 'orig_host2' then host = 'map_host2'; else
*                ...
*
*   Original host       -->      Host to map to
*  -----------------            ------------------
    VXCRNA                        VXCERN
    VXCRNB                        VXCERN

============================================================

* DOMAIN DATA :
* This file contains the mapping of several domains, in connection with
* the SAS job producing statistics about the EAN traffic
* To add another domain mapping, simply edit the file and add the
* relevant line, which will be composed of 2 words (no imbedded blank!) :
* the first one is the name of the domain appearing in the log files
* the second one being the name that you want to appear in the report
*  NOTES : - as you can imagine, each line begining with '*' is a comment
```

```
*          - all the names will be UPPER CASEd anyway )
*          - this file will be processed by the Xedit macro "domain xedi"
*            to produce a file of SAS statements of the following form :
*                if domain = 'orig_dom1' then domain = 'map_dom1'; else
*                if domain = 'orig_dom2' then domain = 'map_dom2'; else
*                ...
*
*   Original domain      -->      Domain to map to
*   -----------------             ------------------
    ARISTOTE                       FR
    BITNET                         EARN/BITNET
    CHUNET                         CH
    DFN                            DE
    DNET                           DECNET
    EARN                           EARN/BITNET
    OZ                             AU
*
* Case of the messages auto-forwarded from Wylbur
*
    REPLY                          FORWARDED

============================================================

* HOSTDOM DATA :
* This file contains the mapping of several domains relative to the value
* of the corresponding host, in connection with
* the SAS job producing statistics about the EAN traffic
* To add another domain mapping, simply edit the file and add the
* relevant line, which will be composed of 2 words (no imbedded blank!) :
* the first one is the name of the host appearing in the log files
* the second one being the name of the domain that should correspond to
* that host, even if the domain in the log is different
*  NOTES : - as you can imagine, each line begining with '*' is a comment
*          - all the names will be UPPER CASEd anyway
*          - this file will be processed by the Xedit macro "host xedit"
*            to produce a file of SAS statements of the following form :
*                if host = 'host1' then domain = 'map_domain1'; else
*                if host = 'host2' then domain = 'map_domain2'; else
*                ...
*
*   Original host        -->      Domain to map to
*   -----------------             ------------------
    CERNVAX                        CERN
    CERNVM                         CERN
    GEN                            CERN
    MINT                           CERN
    CEARN                          CERN
    LEPXS                          CERN

============================================================

* NOTCERN DATA :
* This file contains the names of several domains for which, when an
* address is written like user@smthg.domain.CERN, the address is to
* be remapped to user@smthg.domain, i.e. the .CERN part must be removed
* To add another such domain, simply edit the file and add the
* relevant line, which will be composed of 1 word, which is the name of
* the domain in question.
*  NOTES : - as you can imagine, each line begining with '*' is a comment
*          - all the names will be UPPER CASEd anyway
*          - this file will be processed by the Xedit macro "notcern
*            xedit" to produce a file of SAS statements of the following
*            form :
*          if pos('.'||notcerndom1||'.CERN>',hostdom) ¬= 0
*          then hostdo = substr(domain,1,pos('.CERN>',hostdom)-1)||'>';
*          else if pos('.'||notcerndom2||'.CERN>',hostdom) ¬= 0
*          then hostdom = substr(hostdom,1,pos('.CERN>',hostdom)-1)||'>';
*           ...
*
*   Not a .CERN domain
*   --------------------
    DECNET
    REPLY
    EARN
    BITNET
    UUCP

============================================================

* ALLOWED DATA :
* This file is related to the SAS job producing statistics about the
* EAN traffic.
* It contains the list of those domains who are allowed to use CERN
* gateways.
* To add another allowed domain, edit the file and add the
* relevant line, which will be composed of the name of that domain.
*
*  NOTES : - as you can imagine, each line begining with '*' is a comment
*          - all the names will be UPPER CASEd anyway
*          - this file will be processed by the Xedit macro
*            "allowed xedit" to produce a file of SAS statements
*            of the following form :
*                allowed = FALSE;
*                if dom_org = 'domain1' then allowed = TRUE; else
*                if dom_org = 'domain2' then allowed = TRUE; else
*                ...
*                if dom_org = 'domainj' then allowed = TRUE; else
*                ;
*
* IMPORTANT NOTE : the domain listed here should be the ones AFTER the
*                  remapping of the domain names : for example, it's
*                  EARN/BITNET that must appear here, and not EARN
*                  or BITNET, since both have been remapped to
*                  EARN/BITNET.
*
*  Domains (AFTER remapping !) allowed to use CERN gateways :
*  ----------------------------------------------------------
CERN
CH
DECNET
FORWARDED
FR
```

```
IRIS
OSIRIDE
SUNET
UNINETT

============================================================

* CONFIG DATA :
* This file is related to the SAS job producing statistics about the
* EAN traffic.
* It contains the value of some parameters one could wish to modify
* easily.  These parameters are the following :
*
* (IMPORTANT WARNING : THE ORDER OF THESE PARAMETERS MUST NOT BE
*                      CHANGED but comments (lines with '*' in column 1)
*                      can be added anywhere)
*
* The current year (YYYY)
1988
*
* The number of most popular hosts to keep in a report
40
* The number of .CERN hosts to keep in a report
60

============================================================

* COUNCOST DATA :
* This file contains the different values of the countries (and
* continents) a domain can be in and the corresponding transfer cost
* (in SF per 100 segments)
* This file is related to the SAS job producing statistics
* about the EAN traffic
* To add another coutry - cost pair, simply edit the file and add the
* relevant line, which will be composed of 2 "words" (no imbedded
* blank!) : the first one is the name of the country/continent and the
* second one is the amount of SF that it costs to transfer 100 segments
* in that country/continent
*  NOTES : - as you can imagine, each line begining with '*' is a comment
*          - the country names will be UPPER CASEd anyway
*          - this file will be processed by the Xedit macro
            "councost xedit" to produce a file of SAS statements
            of the following form :
*               if country = 'country1' then Cvdom = 'Cvdom1'; else
*               if country = 'country2' then Cvdom = 'Cvdom2'; else
*               ...
*               if country = 'countryj' then Cvdom = 'Cvdomj'; else
*               ;
            where Cvdom is the cost (SF) per volume (100 segment)
            for all domains in the given coutry
*         - it's assumed that all the countries (continent) appearing
*           in the mapping of domains to countries ("domcount data")
*           have a corresponding entry in this file (with no spelling
*           mistake).  The number of such countries being quite small,
*           this assumption doesn't seem to restrictive...
*         - it's up to you to maintain a file up to data and consistent
*           with the file "councost data" containing the cost per volume
*           and per country
*
*  Country             -->      Cvdom     (cost in SF/100segments)
*  ---------                   ---------
   CERN                          0.00
   EUROPE                        0.07
   SWITZERLAND                   0.01
   USA/CANADA                    0.25
   AUSTRALIA                     0.30
   JAPAN                         0.30

============================================================

/* MSG_COST DATA : */
/* This file explains (to you and to SAS) how to calculate the cost of
   a message send through EAN. It is meant to be included (by SAS itself)
   in the SAS job producing some statistics related to EAN traffic

   let nbytes = number of bytes in the message
       btot   = number of bytes actually transferred (btot > b due to
                overhead : headers, trailers, retransmissions...)
       alpha  = factor taking the overhead into account

   we have : btot = nbytes * alpha

   the cost of the transfer of a message is composed of the cost
   related to the volume transferred and the cost related to the
   time spend to complete the transfer.

   let Cvdom = the cost (in SF) to transfer 100 segments to a given
               domain
       Cvol  = volume cost incurred to transfer btot bytes depending
               on Cvdom

       Cvol   = (btot/64)/100 * Cvdom

   let Cavrg = the average cost (in SF) to transfer something during
               one minute
       Cdur  = duration cost incurred to transfer btot bytes depending
               on a average duration cost Cavrg and an average
               transmission speed of 1 kpbs, i.e 100 bytes/sec

    we have :
       Cdur = (btot/100)/60 * Cavrg

    To summarize :
       totcost = Cvol + Cdur
               = (btot/64)/100 * Cvdom + (btot/100)/60 * Cavrg
               = nbytes * alpha * (Cvdom/6400 + Cavrg/6000)

     where Ctot is the total cost (in SF) incurred to transmit a message
     of nbytes bytes
 */

/*************************************************************************/
```

```
/*   H E R E   I S   T H E   P A R T   T H A T   Y O U   W I L L     */
/*   B E   A B L E   T O   M O D I F Y   T O   R E F I N E   T H E   */
/*   T H E   A P P R O X I M A T I O N                               */
/*********************************************************************/
/*                                                                   */
/************************                                           */
/**/  alpha = 2.5 ;  /**   factor taking the overheag into account    */
/**/  Cavrg = 0.10 ; /**   average cost (in SF) to transfer something */
/************************   during 1 minute                          */
/*                                                                   */
/*********************************************************************/

 /* Cvdom has already been calculated by the preceding SAS macro (in the
    SAS job) */

    totcost = nbytes * alpha * (Cvdom/6400 + Cavrg/6000);


============================================================

/* COUNCOST Xedit : builds a file of SAS conditionnal SAS statement from
   a text data file containing the mapping of domains to countries */
"COMMAND EXTRACT /SIZE"
filesize = size.1

"COMMAND TOP"
"COMMAND DOWN"
do filesize
  "COMMAND STACK"
  "COMMAND DEL"
  parse upper pull mapcount  mapcost  .
  if substr(mapcount,1,1) ¬= '*' then do
     "COMMAND UP"
     "COMMAND I if country = '" || mapcount || "'"
     "COMMAND I then Cvdom = " || mapcost || "; else"
     "COMMAND DOWN"
  end
end
"COMMAND I ;"
"COMMAND FILE"

  ============================================================

/* This SAS job is meant to get from the data sets only that information
   to send through mail for a quick check by the responsible */

CMS FILEDEF CONFIG DISK CONFIG TEST E;      /* gives some config data */

 options pagesize=100 linesize=79;

data temp3;
  set mnthtime.data;
  put type type_nb;

data temp1;
  file print;
  set mnthtime.data;
  put event ' on ' date ' at ' time;


data temp2;
  file print;
  set mnthtype.data;
  if type='export' then put 'Number of messages exported : ' type_nb;


proc plot data=exp_msg.data;
  plot exp_avg*lastdate='*' / vzero vpos=20;
  title Average number of messages exported per day;


data mnthsum2 (keep = hostdomo host_org dom_orig
                      hostdomd host_dst dom_dest
                      nbytes totcost obs_numb)
     mnthdst2 (keep = hostdomd obs_numb);

  format country $20.;
  informat country $20.;

  set monthsum.data;
  output mnthdst2;

  firstdot = index(hostdomo,'.');
  hd = hostdomo;
  do while (index(hd,'.') ¬= 0);
     lastdot = index(hd,'.');
     hd = substr(hd,1,lastdot-1) || '-' || substr(hd,lastdot+1);
  end;
  host_org = substr(hostdomo,1,firstdot-1);
  dom_orig = substr(hostdomo,lastdot+1);

  firstdot = index(hostdomd,'.');
  hd = hostdomd;
  do while (index(hd,'.') ¬= 0);
     lastdot = index(hd,'.');
     hd = substr(hd,1,lastdot-1) || '-' || substr(hd,lastdot+1);
  end;
  host_dst = substr(hostdomd,1,firstdot-1);
  dom_dest = substr(hostdomd,lastdot+1);

  output mnthsum2;

/*********************************************************************
* Construction of data set summarizing the number of messages       *
* received, sent and exchanged by CERN hosts                        *
*********************************************************************/

data CERN_hst (keep = hostdom received sent xchanged);
  set mnthsum2;
  if dom_orig = 'CERN' then do;
       hostdom = hostdomo;
       sent = obs_numb; received = 0; xchanged = obs_numb;
```

```
      output;
   end;
   if dom_dest = 'CERN' then do;
      hostdom = hostdomd;
      received = obs_numb; sent = 0; xchanged = obs_numb;
      output;
   end;

proc summary data = CERN_hst;
  class hostdom;
  var received sent xchanged;
  output out=temp_hst sum=received sent xchanged;

proc sort data = temp_hst;
  by descending xchanged;

data tmp2_hst (keep = host received sent xchanged);
  set temp_hst;
  host = hostdom;

  /* Read the value of nb_cernh (the number of .CERN hosts to list)
     in the config file */
  %include config;

  if _N_ ¬= 1 & _N_ <= nb_cernh+1;


proc print data = tmp2_hst;
  title Number of messages received, sent and exchanged by .CERN hosts;

/*************************************************************************
* Printing the most popular host-domain pairs as destination             *
*************************************************************************/

proc summary data=mnthdst2;
    class hostdomd;
    var obs_numb;
    output out = mnthdst3 sum = obs_numb;

proc sort data=mnthdst3;
    by descending obs_numb;

data temp_dst (keep = dest obs_numb);
    set mnthdst3;
    dest = hostdomd;

    /* Read the value of nb_pop (the number of most popular domains to
       list in the config file */
    %include config;

    if _N_ ¬= 1 & _N_ <= nb_pop+1;

proc print data=temp_dst;
    title Most popular destination hosts;


============================================================

/* This Xedit macro will modify the file it has been called for (the
   file SASMAIL LISTING normally), and get all the line begining with
   "1" out of it before FILing it */

"COMMAND LOCATE :1"
"COMMAND SET ARBCHAR ON *"

i = 0 ; fini = 0
do while ¬ fini

   "COMMAND STACK"
   pull line
   if substr(line,1,2) = '1 ' then do ;
       "COMMAND CHANGE /*//"
       if i ¬= 0 then do
          "COMMAND I -"
          "COMMAND I -"
          "COMMAND I -"
          "COMMAND DOWN"
       end
       cleanline = "-"
       if word(line,2) ¬= 'SAS'
           then do j=2 to words(line) - 1
                  cleanline = cleanline word(line,j)
                end
       "COMMAND CHANGE /*/"cleanline
       "COMMAND DOWN"
       "COMMAND CHANGE /*//"
       i = i + 1
   end
   "COMMAND DOWN"
   if rc ¬= 0 then fini = 1
   i = i + 1
end

"FILE"

exit
check:
"COMMAND STACK"
pull l ; say l
return

============================================================
```

21 January 88

Description of the EAN statistic batch job on CERNVM
==========

To automate the analysis of the log files produced by EAN activity, a batch job resubmitting itself is used. Its function is twofold :

- to analyse the data consisting of the log files produced by EAN

activity
- to produce a report containing the results of the analysis (at requested interval)

Some definitions

(the data and command files mentionned below all can be found on CERNVM, account EMSTAT, minidisk A or D)

- input file : the file containing the EAN log files that the batch job running on VXGIFT has transferred since the statistic batch job was last run (this file was then erased). This file is EANFILE DATA D.

- data files : these files contain data which are likely to be modified from time to time. They include the remappings of host and domain names, the name of the domains allowed to use CERN gateways, the way the cost of the messages are calculated,... Each time the batch jub is run, these files are processed by an editor macro to produce corresponding files of SAS statements which will be included by the SAS interpreter in the SAS program actually processing the EAN data. These files are * DATA D.

- current data sets: these are the SAS data sets containing the information kept from the EAN log files since a report was last produced (or all was manually initialized). These current data sets are sometimes called the monthly data sets (for historical reasons !)

- yearly data sets: These are the SAS data sets containing the information kept from the EAN log files since the last general initialization of all the data sets (with the command INITALL). A report from them can be produced (non destructively) by the command SASYEAR.

- nb_days_to_wait : parameter found in the begining of the file SASBATCH EXEC A and telling how many days to wait before running the batch job again

- nb_times_to_wait: parameter found in the begining of the file SASBATCH EXEC A and telling how many times to only process the data before producing a report

- responsible : the responsible for the batch job is the person which will receive an E-mail message generated by the batch job itself when :
  - a report has been produced (and printed)
  - there was a problem during the execution of the batch job

  The responsible is also supposed to monitor the proper execution of the batch job (in particular, its correct auto resubmission). It's also likely that this person will maintain this application in general, and in particular will modify some parameters. The data files will probably also be updated by the responsible when necessary.

Some details

Each time it wakes up, the batch job calls a routine which will take the input file and ask SAS (Statistical Analysis System) to analyse it and add the data produced (in a compressed form) to the SAS data sets containing the data produced during the previous analyses.

The batch job's file is SASBATCH EXEC. It can be found on minidisk A of the CERNVM account EMSTAT. The source code is written in REXX. A copy of the source code is included in the attachment.

The parameter nb_days_to_wait tells the batch job how many days to wait before waking up.

The batch job produces a printed report of the data it has accumulated so far when it's time to do so. It knows the right time has come by checking the proper parameter (nb_times_to_wait) against the value stored in the files TIMES DATA D. The parameter nb_times_to_wait says how many times to simply analyse the data and add it to the current SAS data sets before producing (and printing) a report.

Each time the batch job is executed, a value is read from the file TIMES DATA D (which contains only that value) telling how many times the batch job has been running since this value was last reinitialised. That value, after having been incremented, is checked against the value of the parameter nb_times_to_wait. If it's less, then the updated value is written back to the file and nothing more happens. If it is equal (or greater, in case of error, change in the parameter value, or something else), then a report concerning the current data in the SAS data sets is produced and printed and a summary of it is sent to the responsible for this job.

Before doing the actual analysis of the input file, the batch job must have access to the minidisk containing the data files (including the input file). If it can't access that disk (the main reason being that the minidisk is currently write locked, for instance if someone is logged on or has DISConnected without having DROPped the minidisk in question), then a warning message is sent to the responsible and the batch is simply resubmitted for the next time.

The input file is the file EANFILE DATA D which contains the EAN log files produced on VXGIFT. Each day (normally), the batch job whose name is EANROOT:[COM]RENAME_LOGS.COM on VXGIFT sends the EAN log file of the previous day to CERNVM and appends it to the file EANFILE DATA D of the account EMSTAT if this file exists or creates that file if it doesn't already exist.

When the analysis of the input file has been done, the file, after having

been copied in another file, is deleted. Keeping the previous version of the input file is just a measure of security, but there is no automatic procedure of recovery in case of problems (sorry !). The only way to use it is to understand what the batch job does, to understand a little of VM/CMS, and of course you also need a problem.

At the end of the execution of the batch job, the latter resubmit itself with the following options. It will have to wake up nb_days to wait days from today, at 7.30 am (to have the latest news since the EAN data file should arrive every day at about 7.00 am). Its class will be class S, class S (this means a maximum of 4 minutes of CPU time), and it will be asked for certain files not to be returned (otherwise, the batch machine returns all the files remaining on its A disk at the end of execution. There is also an option in the batch submission command line specifying that the batch must be run on the CERNVM CPU, not the default SIEMENS CPU, just because CERN has the license to run SAS on the first CPU, not on the second, and that by default, batch jobs are submitted on the SIEMENS CPU.

Details of the routine controling the actual analysis by SAS

The SASWEEKB Rexx routine is in charge of the actual analysis of the data. It has to access the SAS minidisk, request a temporary work minidisk to VM (and do it in such a manner that if there is a temporary shortage of such a resource, it waits a specified number of times during a specified number of minutes before giving up and exiting with an error code which will be used by the calling Rexx routine (SASBATCH) to send a warning message to the responsible). Then, it creates files of SAS statements from the data files containing information such as how to remap host and domain names, how to calculate messages costs and so on. These files will be included in the SAS programs at execution time. Then the SAS program itself is called.

Details of the error/warning routine

The WARNING Rexx routine is called when there is a problem in the batch job (or at least where I thought their could be an error) and is passed the error code from which a simple message explaining the cause of the problem is generated. This message is then sent to a 'responsible' whose name is specified as a constant (easier for you to change when it's a constant !) at the begining of the routine.

How to get yearly statistics ?

The Rexx command SASYEAR produces a report from the data accumulated (and summarized) since the last time the yearly data was initialized (see below). If everything went well, you are asked whether you want the report printed or not.

What's the use of the exec CALLBTCH ?

The Rexx command controlling the execution of the batch job is SASBATCH. But when you have a closer look at the command (re)submitting the batch job (either in the command (re)launching it (SBATCH), or at the end of SASBATCH itself (to resubmit it for the next time)), you can see that the job file which is submitted to the batch machine is CALLBTCH (the command doing this being BATCH SUBMIT (<options>) CALLBTCH). If you investigate further, you can see that the only thing CALLBTCH do is to call SASBATCH (command : EXEC SASBATCH), nothing more. Why not call SASBATCH directly, then ?

The reason is that, at the end of execution, the batch job resubmits itself, which means sends a copy of the file containing the batch commands to the batch machine. The trouble is that the first thing the batch machine does when starting a job is to copy that file on its minidisk A. Thus, if between two executions of the batch job, you modify the file SASBATCH EXEC, the modifications won't be taken into account, since at the end of execution, the batch machine will be resubmited a copy of the file it just executed. Why ? Because even if you have an updated version of SASBATH EXEC on your minidisk, the search order for a file, in particular for SASBATCH EXEC, starts at minidisk A. So, when it's time to send that file to the batch machine for resubmission, the file is found on minidisk A, which contains the older version of SASBATCH EXEC !). Thanks to the intermediate command CALLBTCH, it's this file (CALLBTCH EXEC) which is copied on the batch machine's minidisk A (no problem, this file is never modified !), and it's only at execution time that the file SASBATCH is read (not copied) from your disk. So, you can be sure your latest updates will be taken into account.

How to stop the batch job ?

In case of problems, it may be necessary to stop the batch job, i.e. to "kill" the job which is planned to run in some time from now. This could be the case when there was some problem and you have been warned be the batch job itself. Normally, in such cases, the batch job simply warns you and resubmit itself for the next day. If, after having fixed the problem, you want the batch job to execute immediately, you first have to get rid of its occurence which has already been programmed to run for the next day (in this example).

So, you first have to ask the list of the batch jobs (yours !) which are currently waiting to execute, what you can do be the command BATCH QUERY. Then, the batch monitor answers with the list of those waiting jobs (in this case, there is normally one such job). You spot the ID of the job (which must be something like DHExxx, where xxx is a number), and then you purge the job by BATCH PURGE DHExxx. The batch monitor then kills the job from its queue, and you're ready to read the next section.

How to (re)launch it ?

To launch the batch job the first time (or to relaunch it after some event cause the halting of the resubmission process), use the command SBATCH which will start it properly. You can specify the date (format MM/DD) as first parameter and the time (format hh:mm) as second parameter. If neither is specified, NOW is assumed (not as the parameter, but has the time you want to launch it !). If only the date is specified, the time is defaulted to 7.30 am.

How to initialize the data sets ?

You can selectively initialize (this means clear, reset to zero, erase) the EAN data sets containing either the current data only (Rexx command MNTHINIT), or the yearly data only (Rexx command YEARINIT), or all the EAN data, monthly and yearly (Rexx command INITALL).

Normally, only the INITALL command is supposed to be used, for example at the begining of a new year, to restart the statistics from scratch and forget all about the past. The command MNTHINIT is provided to deal with cases (hopefully not to frequent) like when there has been a problem with the execution of the batch job during this 'period' and that you would like to manually reprocess the data for this period only, without affecting the yearly data. This is however not really recommanded...

By the way, BEWARE of these commands !

What kind of messages are you likely to receive ?

You are likely to receive two kinds of messages from the batch job. The first kind is the message indicating that a report has been produced and giving you a summary of the information you will be able to find detailled in the report (which has normally been printed on the printer of the 2nd floor).

The second kind of messages is error messages. These warn you that something wrong happened during the execution of the batch job. For the moment, these messages are the following :

- problem of work space : to be able to process a large amount of data, SAS needs a lot of memory. This latter is requested from the operating system dynamically, during execution of the batch job, before calling SAS. This memory is requested in the form of an additional (temporary) minidisk, on which SAS can store it's temporary work files. Since the requested ressource (in a large quantity !) is not always available, the batch jobs, if not granted the requested memory, waits for a while (that you can determine by changing the parameter mm in the file SASBATCH EXEC) and then retries until it gets what it asked, or until the limit on the number of retries is reached (limit that you can set by modifying the parameter retries in the file SASWEEKB). Each time it retries, the batch job sends you a message saying it will retry in a while. If the limit of retries is reached, it sends you another message indicating it gives up for today, and will try again tomorrow.

- problem of input file : if the batch job doesn't find a file called EANFILE DATA on the minidisk D of EMSTAT, you receive this message. Apparently, this would be caused by the fact that no log would have been transferred from VXGIFT since the last production of a report (and erasing of the file EANFILE DATA D). Maybe there was also a problem with INTERLINK, or simply the disk was write-locked when INTERLINK tried to transfer the file

- problem of locked minidisk : before doing anything, the batch job must ensure it can access in write mode the minidisk on which it is to write the data sets it will produce. If it's not possible, you will know it quite soon !

==========

List of the files related to the EAN statistic analysis

NB : all these files can be found on the EMSTAT account on CERNVM.

Exec's directly callable

SBATCH	EXEC	A	: launches the batch job
SASYEAR	EXEC	A	: controls the production of a yearly report
MNTHINIT	EXEC	A	: initializes the monthly (current) EAN data sets
YEARINIT	EXEC	A	: initializes the yearly EAN data sets
INITALL	EXEC	A	: initializes all the EAN data sets
YEARINIT	EXEC	A	: initializes the yearly EAN data sets

Exec's called by other exec's

CALLBTCH	EXEC	A	: calls SASBATCH and does nothing more
SASBATCH	EXEC	A	: controls the batch job execution
SASWEEKB	EXEC	A	: controls the analysis of the input data by SAS each time the batch job is called
WARNING	EXEC	A	: warns the responsible that a problem arose during the execution of the batch job

SAS files

SASWEEK	SAS	A	: analyses the input file and produces SAS data sets containing the summarized info
SASMONTH	SAS	A	: produces a report file from the current SAS data sets
SASYEAR	SAS	A	: produces a report file from the yearly SAS data sets
SASMAIL	SAS	A	: produces a summary of the report from the current SAS data sets
MNTHINIT	SAS	A	: initializes the current SAS data sets (originally called monthly)
YEARINIT	SAS	A	: initializes the yearly SAS data sets

Input file (contains the data to analyse)

EANFILE	DATA	D	: EAN log file

Data files (to parametrize the SAS analysis)

DOMCOUNT	DATA	D	: contains the mapping of a domain name to the corresponding superdomain, country and flag telling if it's costing something or not to send a message there
COUNCOST	DATA	D	: contains the cost per volume per country depending on host names
HOST	DATA	D	: contains the remapping of host names
DOMAIN	DATA	D	: contains the remapping of domain names
HOSTDOM	DATA	D	: contains the mapping of domain names
ALLOWED	DATA	D	: contains the names of the domains allowed to use CERN gateways
MSG_COST	DATA	D	: contains the detailed way to calculate the cost associated to each message
CONFIG	DATA	D	: contains miscellaneous parameters like the year and the number of most popular hosts to keep in the reports
TIMES	DATA	D	: contains only one string representing the number of times the batch job has been run since this string value was last reset to 0
EMSTAT	DATADISK	D	: this file is a dummy file which is there just to be able to identify the EMSTAT data disk

Xedit files (to create SAS statement files from the data files)

NB : all these files are used by Xedit to transform the corresponding files to SAS statements included in the SAS job at execution time.

CONFIG	XEDIT	A
SASMAIL	XEDIT	A
YEAR	XEDIT	A
ALLOWED	XEDIT	A
DOMCOUNT	XEDIT	A
COUNCOST	XEDIT	A
HOSTDOM	XEDIT	A
HOST	XEDIT	A
DOMAIN	XEDIT	A

SAS data sets (contain the compressed information in SAS format)

DATA	EXP_MSG	D	: contains the data relative to the progression of the number of average messages exported per day
DATA	MNTHTIME	D	: contains the date and time of the first and last message in the current data sets
DATA	YEARTIME	D	: contains the date and time of the first and last message in the yearly data sets
DATA	MNTHTYPE	D	: contains the types and the associated occurence number for the messages in the current data sets
DATA	YEARTYPE	D	: contains the types and the associated occurence number for the messages in the yearly data sets
DATA	MONTHERR	D	: contains the error messages (address format or domain unknown) found in the last messages processed
DATA	MONTHILL	D	: contains the illegal messages found in the last messages processed
DATA	MONTHSUM	D	: contains the data relative to the last messages processed (in a compressed form)
DATA	YEARSUM	D	: contains the data relative to all the messages processed since the last general initialization

c files

EANBATCH	MEMO	A	: details the working of the batch job